



TECHNISCHE
UNIVERSITÄT
DARMSTADT

PEER-TO-PEER LOCATION-BASED SEARCH:
ENGINEERING A NOVEL PEER-TO-PEER OVERLAY
NETWORK

Dissertationsschrift

in englischer Sprache
vorgelegt und genehmigt
am Fachbereich 18

Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt von

DIPL.-ING. ALEKSANDRA KOVAČEVIĆ

geboren am 21. Juni 1980 in Belgrad, Serbien

zur Erlangung des akademischen Grades
eines Doktor-Ingenieurs (Dr.-Ing.)

Erstreferent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Klaus Wehrle

Tag der Einreichung: 14.07.2009
Tag der Disputation: 17.08.2009

Hochschulkennziffer D17
Darmstadt 2009

Dieses Dokument wird bereitgestellt von tuprints, E-Publishing-Service der
Technischen Universität Darmstadt.
<http://tuprints.ulb.tu-darmstadt.de>
tuprints@ulb.tu-darmstadt.de

Bitte zitieren Sie dieses Dokument als:
URN: urn:nbn:de:tuda-tuprints-18931
URL: <http://tuprints.ulb.tu-darmstadt.de/1893>

Die Veröffentlichung steht unter folgender Creative Commons Lizenz:
Namensnennung – Keine kommerzielle Nutzung – Keine Bearbeitung 2.0 Deutschland



<http://creativecommons.org/licenses/by-nc-nd/2.0/de/>

Посвећено мојим највећим љубавима:
мами, тати, Петрику и бати.

ABSTRACT

Personalization of Internet services is a significant feature and exploiting the users' location brings the most value to it. Location-based services have a wide application range – from emergency, tracking, and navigation services to informational and entertainment services.

In existing centrally managed solutions, the results of location-based search are often incomplete or outdated. Additional information about the searched object (e.g. the menu, facilities, prices) is usually not available, as such a huge amount of data and frequent updates (e.g. the number of free places in restaurant) would overload the server. In a peer-to-peer solution, each peer is responsible for the information about the object it represents, therefore, updating and publishing information is done directly without a single point of failure. It is available to a wide community to join and publish their services, as peer-to-peer systems operate at low costs.

The main goal of this thesis is to *prove the feasibility of engineering a peer-to-peer solution for fully retrievable location-based search*.

Following an engineering approach, we first examine the most used and referred overlays of different types (unstructured, structured, and hybrid). Comparative evaluation identifies the influence of their design decisions on quality aspects such as efficiency, scalability, robustness, and stability. The foundation for the design of our solution is based on the findings from this study.

The resulting overlay, Globase.KOM is a structured superpeer-based overlay in the form of a tree enhanced with interconnections. Superpeers are chosen from publicly reachable, static peers with more capacity, spare bandwidth, and good network connectivity. The world projection is divided into rectangular zones, which do not overlap. Each zone is assigned to a superpeer, located inside this zone. It is responsible for all peers in the zone. Superpeers form the tree, which is based on the subset-relation of their zones.

Further contribution is the clear methodology for evaluating peer-to-peer search overlays, by defining metrics and various workloads that address all crucial quality properties. Additionally, in order to model realistic workloads, we discuss the difference between user behavior in popular file-sharing applications and VoIP applications such as Skype. As an evaluation tool, we select the simulation framework Peerfact-Sim.KOM and extend it to support various geographical distribution of peers on a world map and a location-aware churn model.

The evaluation results prove the efficiency, good load balancing, scalability, robustness, and stability of the system. Query resolution is significantly faster than in related solutions. Additionally, the location-awareness of the overlay results in an efficient mapping of the logical overlay to the physical underlay which reduced total transmission delay and unnecessary underlay traffic. Although uneven load distribution

seems to be an issue due to the tree structure of the overlay, we prove very good load balance due to interconnections and a careful zone formation, which together diminish a higher expected overload of superpeers in the higher tree levels. Our solution scales logarithmically with growing network size. It proves to be robust and stable under simultaneous failures of superpeers in higher tree levels, or in the same branch of the tree, and in the case of frequent querying.

The biggest influence on the quality of our solution is the choice of identifier space, its management, and interconnections. A peer identifier contains the information responsibility and its location. That allows smart selection of interconnections and efficient greedy routing. Interconnections enable bypassing of the superpeers in higher levels of the tree and therefore allow equal load distribution among the superpeers. A fast recovery time and small performance variation under extreme churn and critical failures is to be credited to the various maintenance strategies used in combination. The simulation results are confirmed by a prototype and its applicability is shown in the examples of distributed multimedia communication and future peer-to-peer based collaborative applications.

ZUSAMMENFASSUNG

Durch Personalisierung bringen Internetdienste einen großen Mehrwert, insbesondere wenn dabei der Aufenthaltsort des Nutzers einbezogen wird. Lokationsbasierte Dienste haben ein weites Anwendungsfeld - von Notruf-, Verfolgungs- und Navigationsdiensten zu informierenden und unterhaltenden Diensten.

Die Ergebnisse einer lokationsbasierten Suche existierender zentraler Lösungen sind oft unvollständig und veraltet. Zusätzliche Informationen über das gesuchte Objekt (beispielweise eine Speisekarte, die Ausstattung oder Preise) sind selten verfügbar, da solch große Datenmengen und häufige Aktualisierungen (z.B. die Anzahl freier Plätze in einem Restaurant) einen Server überladen würden. In einer Peer-to-Peer-basierten Lösung ist jeder Peer für Informationen über das repräsentierte Objekt verantwortlich. Informationen werden direkt aktualisiert und angeboten, ohne auf eine zentrale Stelle angewiesen zu sein, bei deren Ausfall das System zusammenbricht. Die niedrigen Betriebskosten von Peer-to-Peer-Systemen machen sie für eine große Nutzerschaft verfügbar.

Das Hauptziel dieser Dissertation ist zu beweisen, dass *das Engineering einer Peer-to-Peer-basierten Lösung für lokationsbasierte vollständige Suche machbar ist*.

Als erster Schritt der Engineering-Vorgehensweise, untersuchen wir die am meisten benutzten und referenzierten Overlay-Netzwerktypen (unstrukturiert, strukturiert und hybrid). Durch vergleichende Evaluationen identifizieren wir den Einfluss von deren Designentscheidungen auf Qualitätsaspekte wie Effizienz, Skalierbarkeit, Robustheit und Stabilität. Die Grundlage für das Design unserer Lösung basiert auf dieser Studie.

Das resultierende Overlay-Netzwerk, Globase.KOM, ist ein strukturiertes Superpeer-basiertes Overlay-Netzwerk in Form eines durch Zwischenverbindungen erweiterten Baumes. Die Superpeers werden unter öffentlich erreichbaren, statischen Peers mit höherer Kapazität, unbenutzter Bandbreite und guter Netzanbindung gewählt. Die Projektion der Weltkarte ist in rechteckige, überlappungsfreie Zonen aufgeteilt. Jede Zone ist einem Superpeer zugeteilt, der sich selbst innerhalb dieser Zone befindet. Dieser ist für alle Peers in der Zone verantwortlich. Die Superpeers formen einen Baum, welcher aus den Teilmengenbeziehungen der Zonen besteht.

Ein weiterer Beitrag dieser Dissertation ist eine klare Evaluationsmethodik zur Untersuchung von Peer-to-Peer-basierten Suchoverlaynetzen, durch die Definition von Metriken und verschiedenen Lastmodellen – die entscheidende Qualitätskriterien adressieren. Für realistische Lastmodelle analysieren wir das Verhalten der Nutzern von VoIP-Anwendungen wie Skype, das sich stark von dem Nutzungsverhalten populärer Dateitauschbörsen unterscheidet. Als Evaluationswerkzeug nutzen wir das Simulationsrahmenwerk PeerfactSim.KOM

und erweitern es, um verschiedene geographische Verteilungen von Peers auf einer Weltkarte und lokationsbewusstes Churn unter diesen zu ermöglichen.

Die Evaluationsergebnisse bestätigen unserem System gute Effizienz, Lastverteilung, Skalierbarkeit, Robustheit und Stabilität. Die Beantwortung von Suchanfragen ist entscheidend schneller als in verwandten Lösungen. Zusätzlich ist durch das Lokationsbewusstsein das Overlay-Netz effizient auf das physikalische Underlay-Netzwerk abgebildet, wodurch Übertragungsverzögerungen und unnötiger Datenverkehr im Underlay-Netz reduziert werden. Obwohl es scheint, als würde die Baumstruktur eine ungleiche Lastverteilung zur Folge haben können, widerlegen unsere Untersuchungen dies. Zwischenverbindungen und ein sorgfältiges Formen von Zonen vermindern eine erhöhte Last auf Superpeers höherer Baumebenen. Unsere Lösung skaliert logarithmisch mit wachsender Netzwerkgröße. Sie erweist sich unter gleichzeitigem Ausfall von Superpeers im gleichen Zweig oder in höheren Ebenen des Baumes, selbst bei häufigen Suchanfragen, als robust und stabil.

Den größten Einfluss auf die Qualität unserer Lösung hat die Wahl des Adressbereichs, dessen Verwaltung und die Zwischenverbindungen. Dem Identifikator eines Peers kann man dessen Lage und den zuständigen Bereich entnehmen. Dies ermöglicht eine elegante Wahl von Zwischenverbindungen und effizientes 'greedy Routing'. Die Zwischenverbindungen erlauben die Superpeers in den höheren Baumebenen zu umgehen und ermöglichen dadurch eine gleichmäßige Lastverteilung unter den Superpeers. Die kurze Wiederherstellungszeit nach Peerausfällen und die geringe Variation der Leistung unter extrem hohem Churn und kritischen Ausfällen ist den diversen Wartungsstrategien zuzuschreiben, die kombiniert benutzt werden. Die Simulationsergebnisse wurden durch einen Prototypen bestätigt. Die Anwendbarkeit unserer Lösung zeigen wir beispielhaft für verteilte multimediale Kommunikation und zukünftige Peer-to-Peer-basierte, kollaborative Anwendungen.

*"Feeling gratitude and not expressing it
is like wrapping a present and not giving it."*

— William A. Ward

ACKNOWLEDGMENTS

One of the things I looked forward to most while writing the thesis was writing the acknowledgments to all of the people who contributed to it.

I am pleased to thank Prof. Ralf Steinmetz for believing in a girl whose voice trembled, on the verge of tears during a disastrous interview presentation. Thank you very much for all of your support, guidance, and encouragement! Or (because you insist on speaking more German for practice) – Vielen Dank für Unterstützung und Ermutigung! I would like to thank my co-supervisor Prof. Klaus Wehrle for his kind help and valuable feedback.

My first year in KOM was made so much easier with the greatest mentor one can have, who generously shares his experiences and ideas. Oli, gigantic thank you for all that you have done! Moving to Germany, adapting and just getting by in first weeks would have been much more complicated and difficult without the huge support and help from Nico. Thank you for all your advice, friendship, frankness and many funny moments. Kalman, thanks for being a fantastic colleague and team worker, full of energy and ideas, always ready to help and sacrifice your time. Most of all thank you for being such a great friend!

My P2P Group@KOM was the source of most of the ideas, motivation, and fun at work. Sebastian, my first and best student, a brilliant colleague, who together with Konstantin provided powerful simulation framework PeerfactSim.KOM - thank you both! Osama, thanks for your support, making us laugh, and providing us with various Lebanese delicacies. Christian and Dominik - thank you for your understanding and support in the last few weeks.

QuaP2P group brought forward many inspiring discussions during three years of cooperation. A special thanks goes to Prof. Andy Schürr, Prof. Max Mühlhäuser, Prof. Alejandro Buchmann, Thorsten Strufe, Dirk Bradler, and Christof Leng for their valuable input.

I enjoy working and brainstorming with my students immensely. A great many thanks go to all of them, especially Leo, Aleksandar, Andre, Till, Pei, Julius, Andre, Hans, Leonid, Nico, Patrick, and Christian.

KOM is a great environment for productive work and personal development. My colleagues Doreen, Sonja, Alex Perez, Stefan Schulte, Julian, Lasse, Parag, Matthias Kropf, Tronje, Mathias Hollick, Michael Niemann, Farid, Andre König, Rainer, Christoph, without really being aware of it, encouraged me with the right words and feedback in the right moments. I would like to thank to Nick for unselfishly sharing his experiences in writing PhD thesis and Andre Miede for this beautiful template. Gisela, Karola, Sabine, Moni, Marion, and Liselotte - thank you for bringing so much warmth and cheerfulness to the institute.

Vasilios Darlagianis introduced me to the peer-to-peer research and taught me how to read perfectly written papers with critical perspective - thank you for everything! I would also like to thank Kai Fischbach, Krishna and Shruti Pandit for all of their valuable hints, encouragement and inspiration. Prof. Abdulmotaleb El Saddik provided valuable tips about research and writing process and learned me the importance of laugh and fun while hard and exhaustive work :)

I am especially grateful to my husband, excellent colleague, and best friend, Patrick for being so supportive, caring and patient with me. Thank you for all of the fruitful research discussions and help in editing the thesis, particularly now in the last weeks of writing. Puno te volim!

Most importantly, I would like to thank my dearest parents for being my best friends as well as my greatest support and inspiration - ХВАЛА за сва ЖРТВОВАЊА за нас. Sale, my coolest and caring brother, thanks for, among all other things, brainwashing me constantly with so many different music styles. Avola is my second family - thanks for all your help. Margret, Salomon, Sonali, Anita, and Christian thank you for being on my side.

A special thank you goes to my dear and supportive friends Irena, Milica, Ljubica, and Ana who proved that distance does not influence real friendship. Bibiana, thank you so much for your support in the first phases of writing this thesis and for being such a dear and caring friend. This thesis would not have any 'a' or 'the' without Annabel Baird, who proof read my thesis. Thank you for being such a cheerful and positive friend and encouraging me with e-mails starting with "Dear Dr. Sandra" :-). Aleksandra Popovic and Zoran Dimitrijevic had a big influence on me five years ago, introducing me to the research world which inspired my decision about my career path.

Beyond being a significant part of my research, Skype and P2P technology allow me stay in contact with my family and friends in Belgrade while Facebook cures occasional nostalgia. Acknowledgments must go to the Matrix Soundtrack which was always there for me during long nights in the office, keeping me running and giving me the illusion of saving the world :-)

CONTENTS

I	INTRODUCTION	1
1	INTRODUCTION	3
1.1	Motivation	4
1.2	Requirements	5
1.2.1	Application Areas	5
1.2.2	Functional Requirements	6
1.2.3	Non-functional Requirements	6
1.3	Challenges	8
1.4	Goals	9
1.5	Outline	10
2	RELATED WORK	11
2.1	Indexing Spatial Data	11
2.1.1	Space Partitioning	11
2.1.2	Using Space Filling Curves	13
2.2	DHT-based Approaches	14
2.3	Tree-based Approaches	15
2.4	Summary	18
II	PEER-TO-PEER LOCATION-BASED SEARCH	19
3	DESIGN OF PEER-TO-PEER OVERLAY NETWORKS TODAY	21
3.1	Taxonomy of the Peer-to-Peer Overlays	21
3.1.1	Purpose	21
3.1.2	Structure	22
3.2	Taxonomy of the Queries	25
3.3	Classification of Design Decisions	26
3.3.1	Choice of an Identifier Space	26
3.3.2	Mapping of Entities to the Identifier Space	27
3.3.3	Management of the Identifier Space	29
3.3.4	Graph Embedding	29
3.3.5	Routing Strategy	31
3.3.6	Maintenance Strategy	34
3.4	Influence of Design Decisions on the Quality	39
3.4.1	Scalability, Efficiency, and Validity	39
3.4.2	Fairness	43
3.4.3	Robustness and Stability	45
3.5	Lessons Learned	46
3.6	Summary	47
4	A PEER-TO-PEER OVERLAY FOR LOCATION-BASED SEARCH	
-	GLOBASE.KOM	49
4.1	Assumptions	49
4.1.1	Geolocation	49
4.1.2	Heterogeneity of Peers	50
4.1.3	Mobility of Peers	51
4.1.4	Description of Resources	51

4.2	A Brief Overview of Globase.KOM Overlay Network	51
4.3	Design Decisions	52
4.3.1	Choice of an Identifier Space	52
4.3.2	Mapping of Entities to the Identifier Space	54
4.3.3	Management of the Identifier Space	55
4.3.4	Graph Embedding	56
4.3.5	Routing Strategy	59
4.3.6	Maintenance Strategy	62
4.4	Bootstrapping	64
4.5	Forming the Zones	65
4.5.1	Finding a Hotspot	66
4.5.2	Resolving Zone Overlapping	67
4.5.3	Extending a Zone	68
4.6	Summary	68
III	EVALUATION	71
5	EVALUATION GOALS AND METHODOLOGY	73
5.1	Goals of the Evaluation	73
5.2	Metrics	75
5.2.1	Basic Metrics	75
5.2.2	Derived Metrics	76
5.3	Workload	78
5.3.1	Experiment Timelines	78
5.3.2	Churn Models	80
5.4	Measuring Quality Properties	81
5.5	Evaluation Techniques	83
5.6	Summary	84
6	EVALUATION RESULTS	87
6.1	Parameters Calibration	87
6.1.1	Load Threshold Parameters	88
6.1.2	Number of Interconnections	95
6.1.3	General Overlay Parameters	99
6.1.4	Summary	99
6.2	Efficiency	99
6.3	Fairness	102
6.4	Scalability	103
6.5	Stability	103
6.5.1	Intensive Leaving	103
6.5.2	Intensive Queries	106
6.6	Robustness	107
6.6.1	Intensive Failures	108
6.6.2	Multiple Failures in the Same Tree Branch	111
6.7	Comparison with the Related Work	111
6.8	Analysis of Requirements Realization	114
6.9	Summary	115
7	PROOF OF CONCEPT	119
7.1	Prototype User Interface	119
7.2	Design	120
7.2.1	Architecture	120

7.2.2	Underlay-Overlay Interface and Network Layer	122
7.2.3	Interface between Overlay and Application Layer	122
7.2.4	Messages	123
7.2.5	Message Processing	124
7.2.6	Routing Tables	125
7.2.7	Events	126
7.2.8	Statistics	126
7.2.9	Daemons	127
7.2.10	Geolocation	127
7.3	Prototype Measurements	129
7.4	Summary	130
IV	FINALE	133
8	CONCLUSION, SUMMARY AND OUTLOOK	135
8.1	Summary and Conclusions	135
8.2	Implications	138
8.2.1	On Distributed Multimedia Communication	138
8.2.2	On Peer-to-Peer Collaborative Applications	141
8.3	Outlook	142
	BIBLIOGRAPHY	145
V	APPENDIX	159
A	DEFINITIONS OF PEER-TO-PEER	161
B	SUPPLEMENTARY DETAILS TO CHAPTER 3: SIMULATION SET- TINGS	165
C	MAP PROJECTIONS	169
D	GEOGRAPHICAL DISTRIBUTION OF PEERS AND LOCATION- AWARE CHURN MODEL	173
D.1	Existing Measurements Studies of Skype	173
D.2	Data Collection	174
D.3	Geographical Distribution of the Supernodes	176
D.4	Diurnal Effect of Supernode Online Behavior	178
D.5	Session Times of Supernodes	180
D.6	Implications on a Churn Model	181
E	A SIMULATION MODEL OF GLOBASE.KOM	183
F	CONFIGURATION OF PEERFACTSIM.KOM	187
G	SUPPLEMENTARY DETAILS TO CHAPTER 6: PARAMETER SET- TINGS	193
H	LIST OF THE MESSAGE TYPES IN GLOBASE.KOM PROTOTYPE	195
	List of Figures	197
	List of Tables	200
	CURRICULUM VITÆ	203
	PUBLICATIONS	207
	ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG	211

Part I

INTRODUCTION

INTRODUCTION

The Internet is an essential means of communication, collaboration, and knowledge exchange nowadays. Trends in Internet usage rapidly changes, ranging from e-mail and browsing the world wide web, to streaming media, Internet telephony, and social networks. Users are not only consuming Internet services, but becoming a part of it by providing various content, from data files, videos, to blogs and social interaction (so called Web 2.0 [O'R05]).

Trends in Internet usage changes rapidly

This trend change shapes the requirements and challenges for the supporting technology. The most widely used Internet communication model is client-server. It is older than the Internet, gaining its popularity as personal computers (the first being Apple II at 1977) made computers available to individuals. In this communication model, there are *clients* that just use Internet services (e-mail, surfing web sites, etc.), and *servers* which offer services (mail, web, or DNS server). The advantages of this model is simple maintenance (e.g. repairs, content update) and control (e.g. access control, content authenticity) of provided services. Bringing Internet to everyone's homes in the middle of '90s, meant more users with greatly varied Internet connection (e.g. dial-up) and capacities (e.g. CPU power, memory); the so called 'edge of the Internet'. Therefore, having a server with high capacity, good network connectivity, and constant availability, certainly alleviates the control of the service quality.

Client-server is the most widely used Internet communication paradigm

Servers provide a service, clients request it

However, with the exponential growth in Internet users (1.5 billion, one hundred times more than in 1995 [Inta]), the number of simultaneous client requests rapidly increases. This can overload a server making it as a resource bottleneck. Scalability issues in this architecture can be solved by adding immense amounts of additional server resources and organizing them in server farms (like at Google [goo]). Even large amounts of data can be efficiently delivered using the client-server approach, i.e. content delivery networks such as Akamai [Aka]. Despite this, the crashes of client/server systems eventually occur under extreme numbers of client requests (e.g. caused by events such as activations of iPhone OS 3.0 [iTuo9] or enormous interest in information about Michael Jackson's death [bbc09]).

Client-server systems are faced with scalability and robustness issues

A different communication paradigm has gained much attention since May 1999, when the file sharing application, Napster was brought to Internet users. Instead of uploading content on servers, to enable other users to download it, Napster users shared their (music) files directly with each other. In that way, users acted as both clients and servers at the same time. This communication paradigm, where participants share their resources (e.g. data, CPU power, memory, bandwidth) without an intermediary server is called *peer-to-peer*. Napster is, how-

In peer-to-peer communication a user is a client and a server simultaneously

ever, not purely peer-to-peer as it relies on an indexing server in order to find a user that has a requested file.

*Popularity of
peer-to-peer increases
due to its low-cost
maintenance and
natural scalability*

Since the vast success of Napster, peer-to-peer gained increasing attention in the research community. Numerous peer-to-peer systems emerged and gained huge popularity among Internet users. Nowadays, the majority of overall Internet traffic is generated by peer-to-peer applications [SM09]. Peer-to-peer makes the system capacity bigger with the growing number of users as it utilizes existing spare user resources (memory, CPU power, data, etc.) and avoids a server overload. Additionally, maintenance of peer-to-peer systems is cheap, as the administration and hardware costs are very low.

*Current peer-to-peer
research is not mature*

However, in spite of all its advantages, peer-to-peer research is still not mature enough to support all changing trends. As peer-to-peer paradigm plays an important role in future Internet applications, a need for support new trends in Internet usage and development of a methodology for the design and evaluation of peer-to-peer systems becomes more apparent.

1.1 MOTIVATION

*Location-based
services are widely
spread*

Location-based services are very important nowadays and have wide application range – from emergency, tracking, and navigation services to informational and entertainment services [MJ05]. Exploiting the users' location brings the most value to personalization of Internet services. Most in-car navigation systems, e.g. iDrive navigator [iDr07], allow the driver to find the nearest restaurant, gas station, shopping center, police station or hospital; the driver's location is determined using a GPS receiver and all information about the object is stored in the navigation software (usually on CD). Desktop location-based search is offered by many online maps like Google Maps [gMa], which provide exact locations of points of interest. Anyone can add their business to the map, and its location and relevant information (website, description) will be retrieved in the search results. In order to help customers find cheap offers in local stores, Slifter [Sli] provides location-based search for products.

*Not only physical
objects, but also
multimedia content
are the subjects of
location-based search*

Physical objects are not the only subjects of location-based searches. According to [Skro7], at least 50% of online news and communities are based on locality. Therefore, many news/blog applications cover locality, like Outside.in [Out], which specializes in location-based blogging that presents current events in the neighborhood [Joh07]. Searched content can therefore range from physical objects and local information (e.g. weather forecasts) to multimedia, related to specific geographical locations (e.g. videocasting or audiocasting of local news, announcement of events, sharing of experiences, and searches for live video-/audiostreams from a given location [LHHS05]).

*Centralized solutions
deliver often
incomplete or
outdated results*

In existing centrally managed solutions, the search results are often incomplete or outdated. For example, when searching for rent-a-car agencies in Darmstadt, the official phone book provides 24 results, while Google maps retrieves 10 and the navigation system from Mercedes

only 5 results. Passing by a gas station while our navigation system shows that the closest gas station is 5 kilometers away, or navigating through a blocked road are frequent occurrences. Moreover, additional information about the searched object (e.g. the menu, facilities, prices) is usually not available as such a huge amount of data and frequent updates (e.g. the number of free places in restaurant) would overload the server.

In a peer-to-peer solution, each peer is responsible for the information about the object it represents, therefore updating and publishing information is done directly, avoiding single point of failure. For example, a computer in a restaurant used for booking and tracking orders can simply join the peer-to-peer system and publish the information interesting to the customers. It can also offer the users a live stream to see the atmosphere. When the restaurant is closed or moves to other location, it is automatically updated by the change of online status or computer location. All changes in prices, menus, special offers will be always up-to-date. As a peer-to-peer systems operate at low costs, it makes them available to a wide community to join and publish their services, unlike storing information on a CD or on centralized server. While the peer-to-peer research community has been very active in the last seven years, current state-of-the-art overlays cannot fulfill the requirements for efficient and fully retrievable location-based search.

Peer-to-peer approach not only fixes these problems, but also brings additional features

Peer-to-peer approach would allow wide community to join and publish various services at low costs

1.2 REQUIREMENTS

After motivating a need for a peer-to-peer approach for location-based search, the requirements for building a suitable overlay are identified in this section. We derive them from application scenarios and distinguish functional, and non-functional requirements.

1.2.1 Application Areas

The most popular application area of location-based search is finding physical objects (such as restaurants, gas stations, hospitals, places of interest) closest to or in the surrounding area of the specified location. Searched content can for example be video- or audiocasting of local news, or live video-/audiostreams from a given location. Another interesting application area is Emergency Call Handling. In order to establish a connection, the corresponding emergency station must first be identified. This information can either be stored centrally, or distributed. The drawback of centralized emergency systems was shown during the 9/11 attacks in the U.S. The emergency systems were overwhelmed with the vast amount of emergency calls occurring after the attacks [Arlo2]. Decentralization of dispatching the calls can solve this issues. Peer-to-peer location-based search can find the appropriate emergency station to dispatch a call, which is usually the closest emergency station or one belonging to predefined responsibility area.

Application areas are searches for physical objects in a surrounding, or finding the appropriate emergency station in emergency call handling

We do not consider search for mobile services, but users can be mobile

In this thesis, we do not consider application scenarios with searching for mobile objects. An example of such an application is a vehicle or package tracking. We assume that users search for static objects - gas stations, restaurants, web cams, emergency stations etc. Support for mobile users must, however be considered in some extent.

1.2.2 Functional Requirements

From the aforementioned application areas, we can define the following functional requirements for peer-to-peer location-based search:

A solution must provide a complete area search

Requirement 1 *Area search for all services or peers in a defined geographical area.*

For the area search, a user specifies a location (which can be its own location or another), radius and a description of the searched object. This description can be a selection of one or more predefined categories (like in navigation systems). This requirement is important for all presented application scenarios. It allows users to define flexible queries in specified surroundings and choose the matching service (e.g. Japanese instead of Italian restaurant or video instead of audio-cast). In emergency call handling, an emergency station can be found by the responsibility areas. Therefore, enabling area search for this scenario is crucial.

A solution must be able to find the closest service

Requirement 2 *Finding a service or a peer geographically **closest** to the specified location.*

This feature is the most used in location-based searches. In scenarios where a user needs a particular service as fast as possible, he will search for the closest located service. For example, finding the closest cash dispensers, gas station or even more importantly the closest hospital. Finding the closest service to any specified location is important to be offered, as we assume that users seek services not only closest to their current location, but also future locations. Finding the closest rent-a-car station to the hotel in holiday destination, or looking for place to dine after the cinema are just some examples where this feature is necessary.

A solution must enable finding a service in the specified location

Requirement 3 *Finding a peer **in** the specific geographical location.*

This feature does not necessarily bring additional value for a user. Users do not know in advance where exactly a object is located or multimedia content tagged to. If they do know it, they do not need location-based search but other location-based services such as navigation. This requirement is, however crucial for realization of the area search or to find closest services in an area which is not user's surrounding.

1.2.3 Non-functional Requirements

The nature of peer-to-peer overlays raise some key issues to be considered

The nature of peer-to-peer overlay networks introduces some key quality aspects each application has to consider. For example, stability or

load balancing are issues raised by the fact that peers, as autonomous entities can randomly leave, join, or perform queries. It results in a large variation of network size, number of exchanged messages, number of stale contacts in routing tables, etc. Furthermore, peers have variable connectivity and failures can appear at a random point of time, which establishes robustness as another important quality aspect. QuaP2P [qua09], a project funded by German Research Foundation, running since 2005, is focused mainly on systematical examination of quality aspects in peer-to-peer systems. Further definitions of quality requirements rely on the findings of this project [Heco6b].

Findings of QuaP2P project used as basis for describing quality aspects

Requirement 4 *Validity, meaning that the query results are complete and correct.*

Query results must be valid – complete and correct

All retrieved results must match the query request (*correctness*). They also must include all matching results existing in the overlay network (*completeness*). When query results are both complete and correct, we say that the peer-to-peer overlay is *fully retrievable*. It is not trivial to fulfill this requirement in the peer-to-peer overlay networks because of the lack of a centralized view on all available resources and services. It describes whether the functional requirements are fulfilled.

Requirement 5 *Efficiency, defined as the ratio of performance and costs.*

A solution must be efficient in terms of performance-costs ratio

Performance refers to overlay operations and service provisioning. Costs are observed from the view of individual peer, the whole peer-to-peer overlay and the IP infrastructure. The definition of efficiency implies that its improvement can be achieved by either increasing the performance and keeping the same costs, decreasing the costs for the same performance results, or both. We cannot claim improved performance without examining the introduced costs.

Requirement 6 *Fairness, meaning that the costs for overlay operations are uniformly distributed over the peers.*

Fairness, meaning that a load must be evenly distributed among the peers

The fair load of a peer should be proportional to its individual capacity. Overload can occur due to the popularity of the services it offers or its more significant role in the overlay. As the load of a peer is part of the previously described costs, the load balancing requirement is a sub-requirement of efficiency. We consider it separately, however, as it describes decentralization and equality of the peers in the overlay which are crucial properties of peer-to-peer paradigm.

Requirement 7 *Scalability, the quantitative adaptability of the overlay to a changing number of participants or services in the overlay, while preserving the validity, efficiency, and load balance.*

A solution must be scalable

Earlier we argued that the peer-to-peer communication paradigm can solve scalability issues client-server solutions have to deal with. However, peer-to-peer overlay networks do not necessarily possess scalability as natural characteristic. If the peer-to-peer overlay introduces a vast amount of traffic to resolve the queries or for maintenance, this traffic overloads the overlay and obstructs it from operating properly.

A solution must be stable under small perturbations

The difference between stability and robustness emphasized

Stability addresses intensive system usage

Robustness addresses failures of system parts

A solution must be robust

Difference between robustness and fault-tolerance emphasized

Many trade-offs between requirements appear

Requirement 8 *Stability, the persistence of the peer-to-peer overlay under system perturbations such as intensive or frequent overlay operations (e.g. queries, leaving, joining).*

The common understanding of stability is not clearly distinguished from robustness. In the following, we make the difference between them, in the scope of peer-to-peer overlay networks. We base it on the definitions of stability and robustness from other fields of science such as physics, applied mathematics, biology, aerospace, and mechanical engineering.

Both stability and robustness describe the behavior of the system under changed conditions. They are concerned with the persistence, or lack thereof, of the specified features under the specified perturbations [Jen03]. We talk about *stability* when perturbations are caused by frequent and intensive system usage (analog to higher speed of airplane). That means, unusually high numbers of requests in a time interval or for the same services, and frequent leaving of peers. We talk about *robustness* when the system changes imply failures of essential parts of the system. Contrary to stability, it is necessary to know the systems' design in order to examine the robustness in the most critical situations (analog to breaking a wing of an airplane).

Requirement 9 *Robustness, the persistence of a peer-to-peer overlay when crucial parts of a system fail.*

This definition of robustness has big similarities to what one can understand under fault-tolerance. These two quality aspects are in fact very similar. Robustness is, however, broader because it examines multiple failures or the failures of the participants identified to have a crucial role in the overlay. Fault-tolerance is a system persistence under single parts failures. The robustness and fault-tolerant system have no single points of failure and repair (system still operates during repairs) and failures do not cause their propagation. A failure in peer-to-peer overlays means disconnection of a peer from the overlay network.

Summarized, a peer-to-peer overlay for location-based search must provide *area search*, finding a *closest* service or a service *on* a specific location. The results of all three types of queries must be *valid* – complete and correct. The overlay must be *efficient*, *scalable*, *load balanced*, *stable*, and *robust*.

1.3 CHALLENGES

There are two main issues that makes designing peer-to-peer overlay fulfill aforementioned requirements challenges. The first of which are numerous trade offs between quality aspects. Second is a lack of systematical engineering and evaluation methodology in designing and evaluation of peer-to-peer overlays.

Designing fully retrievable overlay, i.e. ensuring that all existing matching results will be retrieved, is a challenging task for a decentralized, peer-to-peer approach. Without a centralized view on the offered services and resources, full retrievability can be achieved by an

intelligent greedy routing. This can, however, lead to the generation of too many routing messages which would burden the network. This negatively affects scalability and efficiency. The more peers in the overlay, the more routing messages generated, the bigger the overload until the eventual break of the overlay. Additional messages means increasing traffic on a peer, overlay and underlay layer. With the increased costs, efficiency falls. Additionally to the possible trade-off between retrievability on one side and scalability and efficiency on the other, there are a number of other challenging requirements to be fulfilled at the same time. For example, in order to make an overlay more robust, early failure detection is needed for timely repair. A certain redundancy in the system improves robustness. This brings additional costs for maintenance of replicas and again increase number of exchanged overlay messages. This again affects efficiency and scalability greatly. A careful design which balances all existing trade-offs between identified requirements is a challenging task.

With the lack of a systematical engineering approach, trial & error principle is currently used when designing a new peer-to-peer overlay. The influence of design decisions on the quality aspects of the overlays is unknown, despite a vast amount of existing peer-to-peer solutions. Additionally, comparing the evaluation results from research papers in the field is impossible. There is a lack of commonly used metrics and tests as well as user behavior models. Widely accepted evaluation tools such as NS2 [ns209] for general networking research, do not exist.

The next section uses these facts to derive the main goal and necessary steps to fulfill the identified requirements and overcome these challenges.

Lack of engineering approach in peer-to-peer overlay design and evaluation

1.4 GOALS

The main goal of this thesis is to *prove the feasibility of engineering a peer-to-peer solution for fully retrievable location-based search*. The necessary steps to achieve this goal are:

- *...engineering a peer-to-peer solution...*
 - The design decisions of the existing peer-to-peer overlays should be analyzed and compared. In order to draw some conclusion about the characteristics of overlays caused by the particular design decisions, their influence on all quality aspects must be examined.
- *...fully retrievable location-based search*
 - The foundation for the design of our solution will be based on the findings from the existing design decisions. The design must include all algorithms, mechanisms, and protocols needed to fulfill all previously stated requirements.
- *prove the feasibility...*
 - Evaluation will prove the viability and efficiency of the designed solution. Viability is reflected in stability, robustness,

Learning from the existing peer-to-peer overlay designs

Designing a peer-to-peer overlay for fully retrievable location-based search

Evaluation of the solution and approach

scalability. In order to address the stated requirements, it is necessary to set an appropriate evaluation methodology, select the evaluation tool, and model the realistic workload for the described application areas. A next step is certainly implementation of the simulation model and a reference prototype.

1.5 OUTLINE

Part I: introduction and a review of related work

Part II: the engineering of a novel peer-to-peer overlay for location-based search

Part III: the evaluation of the solution

Part IV: conclusions, implications, and future work

This thesis is organized in four parts.

Part I introduces and deduces the problem statement by this chapter. Chapter 2 gives the classification and overview of existing related work.

Part II is the crucial part of the thesis, presenting the engineering of a novel peer-to-peer overlay for location-based search, Globase.KOM. Chapter 3 gives the analysis of the design decisions in four different types of peer-to-peer overlays. It shows their influence on efficiency, validity, scalability, stability, and robustness. Based on these findings, the basis for our solution is presented at the end of the chapter. Finally the design of the overlay's algorithms and mechanisms is presented in Chapter 4 and each design decision is analyzed.

Part III presents the evaluation of the solution. In Chapter 5, we provide an evaluation methodology and set a set of suitable metrics and workloads for evaluations of each quality aspect. In Chapter 6, we discuss the obtained evaluation results and provides the answer to whether our solution fulfills the requirements stated in this chapter. In Chapter 7, we move beyond simulation, present the implementation of a reference prototype: the proof of concept for our solution.

Finally, in **Part IV** we summarize the thesis and reveal the main findings and conclusions. We show the implications of the presented work, especially in the field of peer-to-peer systems and distributed multimedia communication. New research challenges opened by this work are presented at the end of Chapter 8.

The **appendices** provide the supplementary information to this thesis. Appendix A explains the used terminology. Simulation settings for evaluations presented in Chapter 3 are provided in Appendix B. Supplementary information to the Chapter 4, about the existing map projections are provided in the Appendix C. Experimental measurements on Skype, peer-to-peer VoIP application and captured correlation between churn, geographical location, and distribution of the peers on the map is provided in Appendix D. A simulation model of our solution is presented in Appendix E. Configuration files and detailed simulation settings are given in Appendices F and G, respectively. Details of the protocol messages in prototype implementation are provided in Appendix H.

In this section we present existing peer-to-peer overlay networks for location-based search. Their design goals are often multidimensional range queries or decentralized spatial indexing. Most of them are based on existing spatial data indexing schemes used in centralized databases. In Section 2.1 we describe two indexing structures for spatial data, which are used in peer-to-peer approaches. We can distinguish two main classes of peer-to-peer overlays for location-based search, according to their structure: Approaches based on distributed hash tables (DHT) presented in Section 2.2, and tree-based approaches, presented in Section 2.3. At the end of this chapter, we present the summary (Section 2.4) of the related work regarding the requirements stated in the previous chapter (Section 1.2).

Two classes of related work: DHT-based and tree-based approaches

2.1 INDEXING SPATIAL DATA

Two main classes of indexing structure used in existing peer-to-peer location-based search overlays are *space partitioning trees* and *space filling curves*. They will be explained in the following.

Related works use space filling curves or space partitioning trees for their indexing structure

2.1.1 Space Partitioning

Space partitioning is a process for dividing a space into disjoint subsets [BCKO08]. It is a field of computational geometry especially important in computer graphics and integrated circuit design. Most of the space partitioning methods are recursive, resulting in a hierarchical structure of the regions in space (regions in already existing regions). Relations between the regions can be presented in the form of a tree, which is called a *space-partitioning tree*. As the focus of this thesis is two-dimensional queries, we will focus here on the space partitioning methods for planes. Here we will present *binary space partitioning trees* (BSP), *R-Trees*, and *KD-Trees*, as they are adopted in peer-to-peer overlays.

Space partitioning methods divide space recursively, resulting in a hierarchical structure (a tree) of the regions

The most commonly used space partitioning method is *binary space partitioning tree* [FKN80]. It divides a plane recursively into two until certain partitioning requirements are satisfied (e.g. all regions must be convex). Figure 1 illustrates this method on a rectangular region. Node A in the tree corresponds to the entire plane. It is connected with the nodes B and C which correspond to the first two regions formed by splitting the plane into two.

Binary space partitioning tree halves regions recursively

The resulting tree is optimal if it is balanced, which is sometimes opposed to the splitting algorithm. It is mostly used in collision detection in computer games (e.g. Doom was the first to use BSPs) or ray tracing.

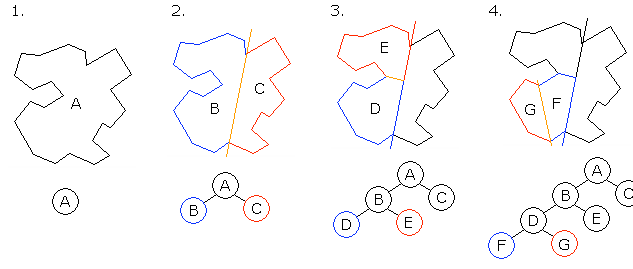


Figure 1: Example binary space partitioning

KD-Trees' splitting lines must be perpendicular to the coordinate axis; a tree node corresponds to a point in a region

A *KD-Tree* [Ben75] is a special type of BSP, where the splitting planes (in two dimensional space, splitting lines) must be perpendicular to the coordinate axis.

Additionally, a node in the tree corresponds to a point in a region, unlike a node in BSP where nodes correspond to any geometrical form. This space partitioning method divides the space into two subspaces, such that each point in space is contained in its own region, if possible. Figure 2 illustrates a KD-Tree. The first splitting line is parallel to x

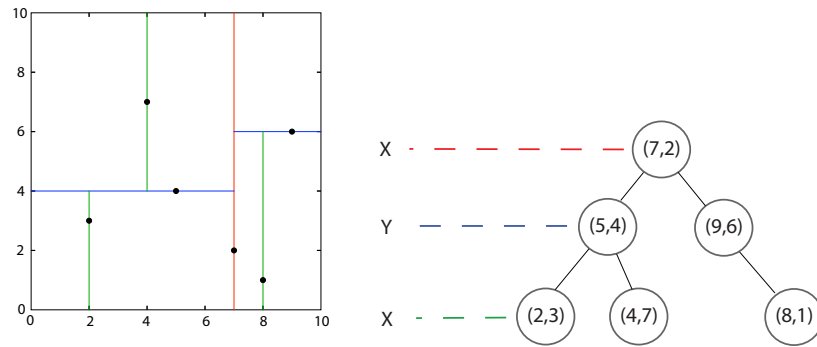


Figure 2: KD-Tree

axis (parked on a first level of a tree in Figure on right side), caused by a point (7,2), which is then assigned to the root in a tree. The next two splits are for points (5,4) and (9,6) parallel to y axis and they are assigned to the separate nodes. The last splitting lines are parallel to x axis, for points (2,3), (4,7), and (8,1).

R-trees form hierarchically nested minimum bounding rectangles which may overlap

A *R-trees* does not divide the entire space but form hierarchically nested minimum bounding rectangles (MBRs) which may overlap. Its application area is spatial queries defined in this thesis as *area search*, as requirement 1 (find all objects in a 5km surrounding). An R-tree is height-balanced (keeping the height constant). Mapping the formed regions and their relations in T-Tree is illustrated in Figure 3.

Each node in an R-Tree corresponds to the smallest MBR that surrounds its children in the tree. As the MBRs can overlap and one rectangle can belong to several MBRs (like region R6 in Figure 3), sev-

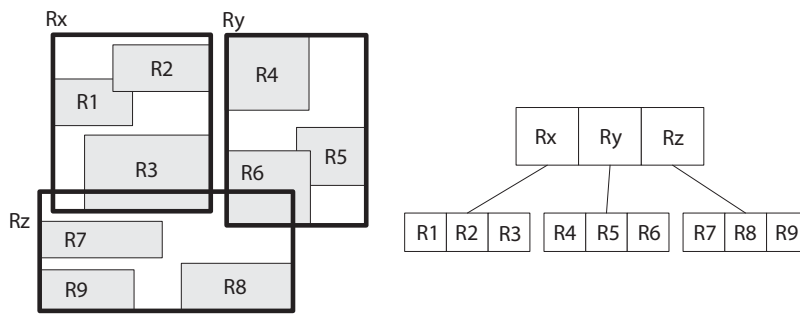


Figure 3: R-Tree

eral nodes in different branches must be visited in order to tell if the searched rectangle exists.

2.1.2 Using Space Filling Curves

A space filling curve is a parameterized, injective function that maps a unit line segment to a continuous curve in the unit square, cube, hypercube, etc, which gets arbitrarily close to a given point in the unit cube as the parameter increases. As it maps two or more dimensional spaces into one-dimensional lines, it is used for dimension reduction and sparse multi-dimensional database indexing [LK00].

Space filling curves map two or more dimensional spaces into one-dimensional lines

For the purposes of spatial queries, the goal is to find the curve with the best locality property, meaning that the closeness of two points in a space will reflect on their distance on a curve. However, “it is impossible to create a space filling curve, such that all points close in multi-dimensional space, are close on the curve, too.” [GL96]

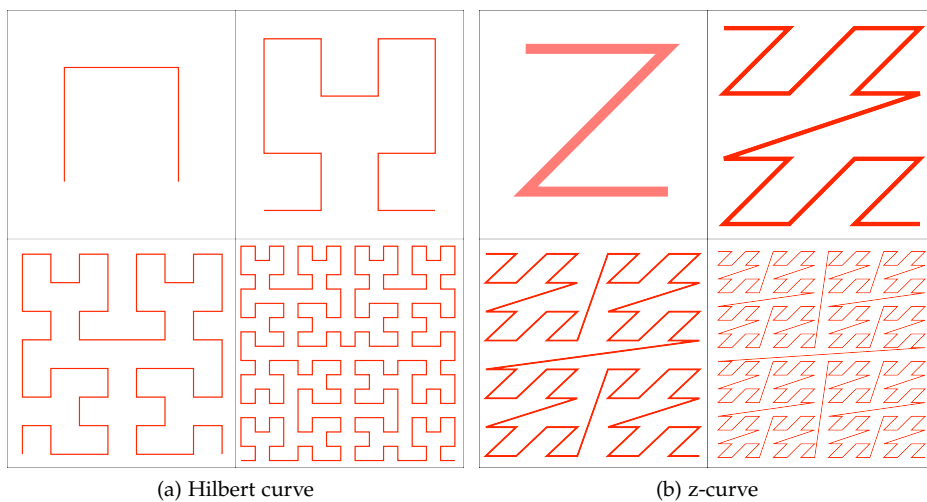


Figure 4: Four iterations of Hilbert and z- space filling curves

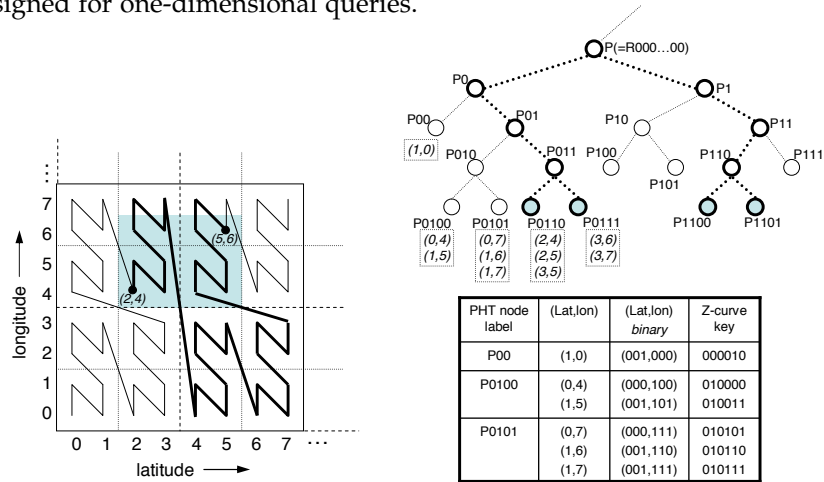
Figure 4 shows Hilbert and z- space filling curves. It represents four iterations of both curves, where the increased curve parameter (i.e. iteration number), means that more space points are covered by a curve. Mapping of two dimensional coordinates to a point in a z-curve is done by interleaving the bits of the binary representation of the x and y coordinates. E.g. the point (X,Y) represented in a binary form as $x_1x_2x_3x_4$ and $y_1y_2y_3y_4$, corresponds to the $x_1y_1x_2y_2x_3y_3x_4y_4$ value on a z-curve. As it is very simple to understand and use, z-curve is often used. Hilbert curve has, however, better locality properties and is advised to be used for multidimensional indexes. Knoll et al. discuss in [KW06] the suitability of various space filling curves for spatial queries.

2.2 DHT-BASED APPROACHES

Reusing existing overlays with one-dimensional index for two dimensional queries

Location-based search in peer-to-peer overlay networks has been approached by re-using existing overlays that are designed to provide efficient one dimensional queries. Well-established overlays, such as Chord [SMK⁺01] or Pastry [RD01] are used, which implement distributed hash tables (DHTs) to address stored values and efficiently resolve *lookup* queries. More details on distributed hash tables are given in Section 3.1.2. Reusing mature and tested overlays and building new overlays on top might significantly alleviate the task of developing a system from the scratch.

Approaches such as [ZSLSo6, SLLY08, KSSoga] use space-filling curves in order to apply the indexing and routing from overlays designed for one-dimensional queries.



(a) The marked area is the subject of a two-dimensional range query. The marked line represents the corresponding range in a z-curve
(b) Routing performed in PHT, to get the stored items on the leaf nodes

Figure 5: Area search using Prefix Hash Tables (Source: [CRR⁺05])

The goal of [CRR⁺05] to develop Prefix Hash Trees (PHTs) was to meet the needs of PlaceLab's end-user positioning system [LCC⁺05b] without modifying the underlying DHT. The PHT is able to perform

two dimensional geographical range queries by applying a z-curve on a two dimensional space. Additional tree-based indexing data structure is used over existing DHT. Keys are stored at the leaf nodes which have the longest matching prefix. An example for the area search (or two dimensional range query) in a PHT for the points in the space between (2,4) and (5,6) is illustrated in Figure 5a. The corresponding range of the z-curve is marked on Figure 5a. We can see that part of the marked range does not intersect with the searched area. Results of the search are stored at the leaves P0110 and P0111. All contacted peers are marked in Figure 5b. We can see that the query messages are routed to the tree nodes with the longest matching prefix to the z-curve key. In [GGS03, ZGS03] location-based node identifiers are used and mapped onto the two dimensional space using space filling curves as well. Other works that are based on similar transformations of the two dimensional space onto a one dimensional space are [WSS05, BAS04].

GeoPeer is a location-aware P2P system [AR04], which uses Delaunay triangulation to build a connected lattice of nodes and it implements a DHT for geographical routing, similar to GHT [RKY⁺02]. Neither provide support for complete retrievable search. All approaches with space-filling curves are detrimentally affected by the fact that they do not match the geographical distance with the distance in the overlay identifier space. An example is shown in Figure 5a, in which two points are very close geographically but have an inappropriate distance in the overlay. This results in inefficient query replies, which introduce additional delay into the communication because of increased routing hop count. Another important issue is that DHTs are designed for lookups, not for exhaustive searches. Therefore, they cannot provide a guarantee to retrieve all results matching an area search query (Requirement 4).

Prefix hash trees are built on a top of DHT, performing 2D range queries by using a z-curve

DHT-based approaches suffer from poor underlay-awareness and do not provide full retrievability

2.3 TREE-BASED APPROACHES

The other class of peer-to-peer overlay networks designed for location-based search are tree-based approaches. The indexing structure is not one dimensional space, like in DHTs, but two dimensional, using space partitioning methods (see Section 2.1.1).

The first peer-to-peer overlay that supported multidimensional queries was Content Addressable Network (CAN) [RFH⁺01]. It was designed as an implementation of a DHT, but it can be used for spatial queries without much modification. Its indexing structure is similar to KD-trees – it is binary space partitioning, with the splitting parallel to the axis, but in the equal regions. The biggest issue in using CAN for two dimensional spatial queries is its routing complexity $O(d \cdot N^{\frac{1}{d}})$, where d is the number of dimensions, which is very complex for two dimensional space. A similar approach to spatial queries was presented in [HT03, THS⁺04]. Harwood and Tanin recursively divided a two dimensional space into smaller zones and used a distributed quadtree [FB74] index for assigning responsibilities for regions of space to peers. For each zone a control point is assigned and hashed into the node identifier space in a Chord ring. Copies of objects associated with a

Tree-based approaches use 2D indexing structure, mostly by using space partitioning trees

CAN can support multidimensional queries, but has poor routing complexity for 2D queries

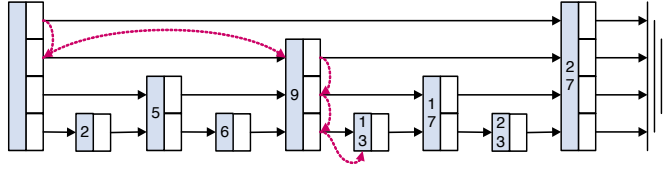


Figure 6: Example of 4-level Skip List with the routing path to the key 13

region are stored on the node which is the assigned control point. As a result, the two dimensional space is transformed into a tree structure.

Zimmermann et al. discuss in [ZKW04, WZK05] that such an approach can lead to load balancing problems. Therefore, they introduced a mapping of the physical space into the CAN [RFH⁺01] overlay instead of Chord [SMK⁺01]. The identifier of indexed spatial data consists of the respective location, a random part and the identification of the content of the object. Similar work is presented in [ZKW05]. Brushwood, based on KD-trees [Ben75] and the search space, is repeatedly hierarchically partitioned into smaller zones and each internal node is split into two subzones. The indexed points are stored in leaf nodes. Points of the search area can be in multiple leaves of different branches, as shown in Section 2.1.1. Thus, a query has to be propagated to the nodes close to root of the tree so that a performance bottleneck at the higher level nodes can occur.

SkipIndex [ZK04] is based on Brushwood and uses a skip graph [AS03] for routing. First it builds a KD-tree to maintain the index and then a skip graph on the leaves of KD-tree to shorten the routing path. An example of a four level skip list, for routing to the key 13 is presented in Figure 6.

P2PR-tree [MLK04] implements R-tree in a peer-to-peer overlay network. Despite its full decentralization, tree levels and corresponding nodes do not have an equal role in indexing and routing. The first two tree levels correspond to the fixed, pre-formed regions while the lower tree levels dynamically grow depending on the number of peers in each pre-defined region. Due to its static space partitioning, the performance of P2PR-tree [MLK04] might be poor when used for location-based search, as the peers' distribution is not uniform in real world conditions. Additionally, space partitioning based on R-Tree allow overlapping, meaning that the searched region can be stored at multiple branches (see Section 2.1.1) resulting in expensive searches.

MURK (Multidimensional Rectangulation with KD-trees) is similar to CAN in space partitioning and routing strategy. However, it splits the load into equal halves rather than the space. Its routing structure is not tree-based, but it uses spatial neighbors, like in CAN, until it reaches the destination. It, therefore inherits poor routing performance for two dimensional queries.

LL-Net [KHF⁺05] uses a flexible space partitioning similar to R-Trees, but without overlapping. It assigns a so called R-Peer to each region as the root of the tree topology formed the N-Peers contained in that

An approach uses location-aware identifiers to map space partitions on CAN nodes

P2PR-tree might perform poorly when the peers' distribution is not uniform and have an expensive query

MURK: splitting space into equally loaded regions instead of spatial halves

region. It uses a central instance, the S-Peer, which manages the connection to all R-Peers, inheriting the drawbacks of central management.

A recent peer-to-peer overlay for location-based search, inspired by our approach is GeoP2P [AvBo9]. It uses similar zone forming to our approach, but the overlay is not hybrid. Routing strategy and routing table are similar to that in Pastry, with routing tables' entries corresponding to the levels and distances in a tree.

RectNet [Heu05] is a distributed space partitioning tree (see Figure 7) which is the most comparable to our solution. It is based on a KD-tree where all indexing points are at the leaves of a tree. Similar to MURK it dynamically adapts to the geographical distribution of the workload caused by the storage of (location, object)-pairs and the processing of queries. Routing, however relies heavily on the tree structure. We chose this peer-to-peer overlay as a reference for comparative evaluation as it overcomes many issues DHT-based solutions have and has better routing performance than CAN-like approaches. Each peer has a position

RectNet is based on KD-Tree, all indexing points are at the leaves

We chose RectNet as reference overlay

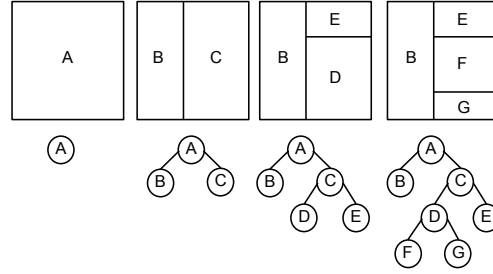


Figure 7: Illustration of the RectNet overlay structure

in a two-dimensional space. When the first node joins the network, it becomes a so-called clusterhead. Clusterheads are organizational units of the overlay. They take responsibility for a certain region of the two dimensional space, meaning the peers and other sub-clusterheads located in this region. The first clusterhead of the first peer takes the responsibility of the entire space. Other joining peers become children of this clusterhead. When the number of children of a clusterhead A exceeds a certain value (maximum load), the region of responsibility is split into two sub-regions. For this purpose, two peers B and C get a role of new clusterheads. The splitting is performed in a way which guarantees that the two sub-zones contain almost an equal number of peers (similar to MURK). The intention behind this is to achieve good load balancing between the two newly created clusterheads, assuming that each peer produces the same amount of traffic. The joining procedure is illustrated in Figure 7. The clusterhead A is referred to as parent clusterhead of B and C. In the first iteration, presented in the first of four figures, the clusterhead A has contacts to all indexed peers in the entire space. When its load exceeds a maximum load value, it splits its space into two regions and assigns them to clusterheads B and C (second figure). Clusterhead A then has no further information about spatial index (other peers in its region) than the connection to clusterheads B and C. All peers in the left region are assigned to clusterhead

B and all peers from the right region are assigned to clusterhead C. At the end, information about the peers in the regions are stored only at the leaves of the tree.

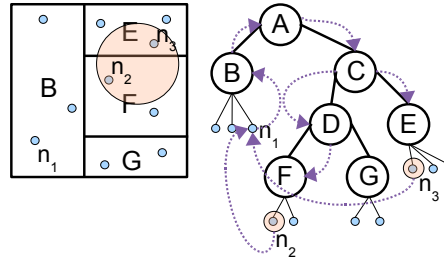


Figure 8: Example of an area search. Node n_1 is the initiator of the search.

An area search query is presented in Figure 8. The peer n_1 sends a query request to its parent B. As B's region does not cover the search area, B forwards the request to its parent. This process is repeated until a clusterhead is reached which covers the whole search area (in our example, clusterhead A). The query message is then forwarded to the child clusterheads, whose regions intersect with the search area. The request is passed down the tree until a leaf-clusterhead is reached which has information about the peers in the searched area. In the given example, this applies for clusterhead E and F. These clusterheads send a result message to n_1 . The result message contains the contact information of all peers which are located in the search area.

The binary structure of the tree simplifies recovery upon peer failures as the number of sub-clusterheads is known (always two). However, a query usually needs to be forwarded to the root and then to the leaves which might affect search performance. In Section 6.7, we evaluate this overlay and compare it to our solution.

2.4 SUMMARY

Current peer-to-peer overlays for location based search are based on either existing DHT implementations or use tree-based indexing or routing. As an indexing structure, space-filling curves are mainly used to map a two dimensional space into a one dimensional curve in order to reuse DHTs that are designed for one dimensional lookups. DHT-based solutions are not fully retrievable and have inadequate mapping between logical overlay structure and underlying network. Other approaches use space partitioning trees like KD-trees or R-trees. Routing in tree-based approaches is very expensive, not greedy, and introduces many unnecessary messages. It increases the response time and protocol overhead. Additionally, loads are most often not evenly distributed among the peers.

RectNet might have search performance issues, but has simple failure recovery

DHT-based approaches are not fully retrievable and often have poor underlay-awareness

Tree-based approaches involve expensive routing and have load balancing issues

Part II

PEER-TO-PEER LOCATION-BASED SEARCH

Engineering means acquiring and applying the knowledge about existing design decisions from the relevant fields to find a suitable solution which realizes a desired goal. As already stated in Section 1.4, the main goal of this thesis is to *prove the feasibility of engineering a peer-to-peer solution for fully retrievable location-based search*. We, therefore, examined the most used and referred to overlays of different types (unstructured, structured, and hybrid). Firstly we present the taxonomy of peer-to-peer overlay networks in Section 3.1. We distinguish three criteria for the taxonomy: purpose, structure and query. We selected Chord, Kademia, Gia, and Gnutella 0.6 to represent different structure types of peer-to-peer overlay networks. In Section 3.3, we analyze their designs, with six classes of design decisions according to the reference architecture model of peer-to-peer overlays from Aberer et al [LAG⁺05]. We examine the influence of these design decisions on the quality aspects, such as efficiency, scalability, robustness, and stability in Section 3.4. Section 3.5 presents our main findings and provide the basis for the design of our solution regarding the stated requirements.

Following an engineering approach, we first examine the most used and referred to overlays of different types

3.1 TAXONOMY OF THE PEER-TO-PEER OVERLAYS

Since the huge success of Napster, many new peer-to-peer overlay networks have been proposed. Although most of them are built for file-sharing, there are many more application areas for the use of peer-to-peer overlays are used. From centralized indexing (like that in Napster) to full decentralization, the structure of the overlays varies. Additionally, there are many types of queries that users or applications can perform on these overlays. Therefore, we present three taxonomies according to the purpose, structure and query type of the peer-to-peer overlay network.

We classify peer-to-peer overlays according to their purpose, structure, and types of queries they support

3.1.1 Purpose

A peer-to-peer overlays have a variety of applications and aims. Nevertheless, we can distinguish the main classes of the existing overlays according to their purpose:

- **Search overlays:** These overlays allow peers to publish and query resources stored in the overlay network. This is an essential functionality for peer-to-peer file sharing. As most of the overlays are deployed for this purpose, they will be the main focus of this chapter. The taxonomy of the queries performed in search overlays is described in Section 3.2.

Search overlays provide decentralized indexing

Content distribution overlays enable content download from multiple sources, by sharing downloaded parts

Streaming content overlays transmit a continuous flow of multimedia data from and to one or more peers

Monitoring overlays provide a global view on an underlying peer-to-peer overlay

- **Content distribution:** The main aim of these overlays is efficient and scalable delivery of large content. It reduces the traffic load of a single resource provider by allowing users to download the parts (so called *chunks*) of content from one another. The most prominent example for content distribution is *BitTorrent*.
- **Streaming content:** In this class of overlays, a continuous flow of multimedia data is transmitted to one or many peers. Examples are Voice-over-IP (VoIP), IP television or Internet radio. For a unicast streaming session (e.g. VoIP), two peers can easily locate each other using, for example, a DHT overlay and build up a simple transport layer connection for streaming. Additional problems arise with multicast streaming, essential for live streaming, television or radio. The objective of the overlay is to reduce the sender's traffic. The sender would otherwise need to send individual streams to each client (like in a client-server approach). Usually a peer relays the stream for other peers. The challenge is to provide a timely replacement of a stream source in the case of failure.
- **Monitoring of other Overlays:** The lack of central instance in peer-to-peer overlays raises the issue of the control over the system performance and costs. Monitoring overlays are usually built on top of the monitored peer-to-peer overlay. They can collect information about the various performance metrics (query response time, query success), costs (bandwidth) or user behavior in the underlying overlay. The collected information can be used for the self-optimization of the overlay (e.g. setting system-wide parameter values or replacing weak peers) [GKXSo8a].

This taxonomy does not claim to be complete, but provides a brief overview of the main application areas of peer-to-peer overlay networks. Collaborative environments, social networks, and networked virtual environments are just some of the additional application areas covered by peer-to-peer overlays nowadays.

3.1.2 Structure

As previously mentioned, we focus in this chapter on search overlays. In the following, we present a commonly used classification of search overlays according to their topology, routing, roles of the peers in the overlays, and resource management.

This taxonomy focuses on search overlays

Centralized: content transfer is decentralized, indexing is centralized

Pure: no centralized party involved

- In the **centralized** approach, indexing is performed by a single instance. Content transfer is carried out in a decentralized fashion, among peers in the overlay. A popular example of the centralized approach is *Napster*.
- In a **pure** peer-to-peer overlay, each peer has an equal role and no central party is required for the overlay operations. According to the way an overlay distributes the resource management

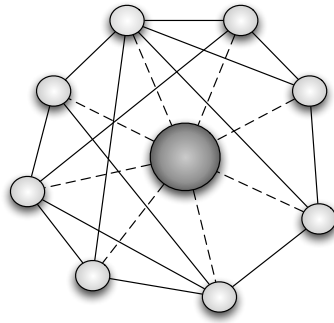


Figure 9: Centralized peer-to-peer overlay network

task among participating peers, we distinguish structured and unstructured overlays:

- In **structured** overlays, there is a strict relation between the peers and the resources they manage. When data is published, an appropriate peer (e.g. one whose identifier value corresponds to a certain function of the resource identifier) or a small subset of peers is chosen to manage it. That means that they retain the reference to that data and make sure it is available. When a peer searches for a resource, it knows the peer identifier that manages the searched resource. Finding a resource then means routing to the particular peer, that holds the reference to it. A neighborhood is selected to enable greedy routing – it is built based on the identifier of a peer and its distance to the potential neighbor. The first structured overlays were implementations of distributed hash tables (DHT). The main purpose is to enable a lookup for a given key $lookup(key)$. The routing in these overlays is based on *Key-based Routing (KBR)* [DZD⁺03], meaning that the selection of the neighbor to which a lookup query is to be forwarded, is based on the relation between its identifier and a key value. We observed *Kademlia* [MM02] as the most used in the deployed peer-to-peer systems and *Chord* [SMK⁺01] as the most referenced overlay. In spite of the fact that both implement a DHT, their designs vary greatly.

Structured: a reference to a resource is placed at the specific peer

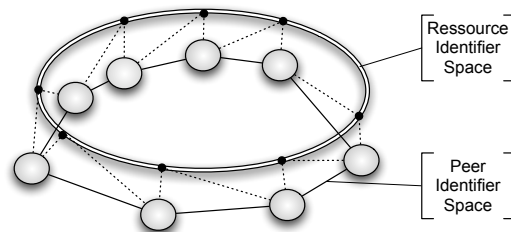


Figure 10: Structured pure peer-to-peer overlay network

Unstructured: a resource is placed randomly, at peer

- In an **unstructured** overlay, there is no relation between an identifier of a resource and an identifier of the peer that manages it. A resource can be managed by any peer. A search for a particular resource is done by flooding – forwarding the search query to all neighbors who do the same if they do not manage the searched resource. Unstructured overlays are often faced with issues of large bandwidth consumption or poor retrievability of rare resources in a large-scale network, as the routing mechanism is random rather than greedy. The first pure peer-to-peer overlay network was Gnutella 0.4 and it was unstructured. As it had huge scalability issues, improved unstructured overlays were proposed to overcome these issues. The direct successor of Gnutella 0.4 was a hybrid overlay, Gnutella 0.6, where the scalability issues are addressed by assigning more important roles to the peers with higher capacities. In this chapter, we observed *Gia* as a representative of unstructured overlays, which tries to compensate scalability issues with an advanced topology management in a purely decentralized manner.

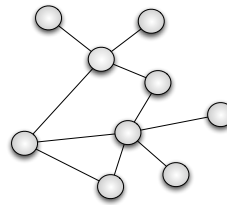


Figure 11: Unstructured pure peer-to-peer overlay network

Hybrid: certain peers have a more important role

- A **hybrid** overlay combines client/server and peer-to-peer communication paradigms. In the Gnutella 0.6 hybrid overlay, peers with high bandwidth capacity (so called *ultrapeers*) communicate amongst themselves in peer-to-peer fashion. Regular peers, with low or normal bandwidth capacity (so called *leaves*) communicate with each other through the corresponding ultrapeer (in client/server fashion). A leaf forwards a search request to its ultrapeer, which finds an ultrapeer assigned to the leaf who stores the requested resource. This reduces the workload for low-bandwidth peers. As a representative of hybrid overlay, we observed *Gnutella 0.6* because it is the most similar to the overlay structure used in popular peer-to-peer applications, KaZaA [KaZ], and Skype [Skyb]. Gnutella 0.6 has numerous implementations, and in this chapter we analyze and evaluate the LimeWire implementation.
- **Hierarchical** overlays use the routing strategy in two or more hierarchy levels. Peers are organized in groups and each group build a pure peer-to-peer overlay. Each group has at least one *group head*, a peer which represents this group. A top-level-overlay connects all groups by connecting their group heads. Short-range

Hierarchical: consist of two or more overlays and routing is done in several levels

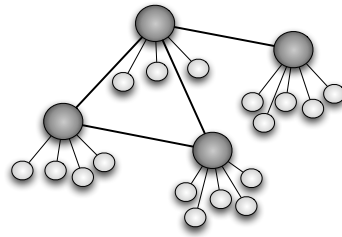


Figure 12: Hybrid peer-to-peer overlay network

queries are resolved within the group, while long-range queries are resolved by contacting the group head which routes the query to the group of final destination. The goal of the hierarchical approach is to reduce the overhead in large overlays, especially in the case of short range queries. It usually combines unstructured and structured overlays.

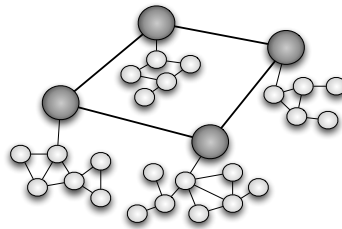


Figure 13: Hierarchical peer-to-peer overlay network

In this chapter, we analyze pure and hybrid overlays. Centralized approaches are not of interest to us as the decentralized indexing of spatial data is the main focus of this thesis. As hierarchical overlays are not widely used or deployed, we do not observe them in this chapter. Additionally, their design is based on the “flat” peer-to-peer overlays. We believe, however, that with careful combination of two overlay types and parameter choices, the drawbacks of each type can be eliminated while inheriting the benefits from the hierarchical approach.

We analyze pure and hybrid overlays

3.2 TAXONOMY OF THE QUERIES

This thesis focuses on search overlays. In DHTs, a user performs a **lookup** for a value by a given key (its hash), while in an unstructured overlay it performs a **search** for a file by its name or its description. When a searched resource is queried by giving key words, we call it **key-based search**. In VoIP applications, a user needs to find a particular peer, which we refer as **find a peer**. The difference between lookup or search for a particular file and finding a peer is that the latter has a single possible destination. In lookup for value, a query can end up at a peer who cached a searched resource. A user can perform **range queries** for the resources whose description or identifier value

Query types: lookup, search, find a peer, range queries, location-based search, etc.

is between a given upper and lower border (e.g. ‘find all publications that appeared between 2005 and 2009’). When more ranges in query criteria are given (e.g. ‘find all publications appeared between 2005 and 2009 that were cited between 50 and 1000 times’), we call it **multi-dimensional range query**. The focus of this thesis is **location-based search** which is a generalization of two-dimensional range queries, as the shape of the searched area does not have to be rectangular (which is the case in two-dimensional range queries).

3.3 CLASSIFICATION OF DESIGN DECISIONS

We analyze structured (Chord, Kademlia), hybrid (Gnutella 0.6), and unstructured (Gia) overlays

In this section we analyze the selected overlays (Chord, Kademlia, Gnutella 0.6, and Gia) by discussing their design decisions. We used the classification of key design aspects defined by Aberer et al. in [LAG⁺05], who proposed a reference architecture for overlay networks. This way, we can better compare overlays’ design decisions and assess the influence they have on quality aspects. Six key design aspects in peer-to-peer overlay networks identified in [LAG⁺05] are:

Six key design aspects in peer-to-peer overlays

- **Choice of an Identifier Space:** describes the space of identifiers for each peer and resource in the network. Furthermore, it defines a closeness metric between the identifiers which can be used for routing through structured overlays.
- **Mapping to the Identifier Space:** defines how identifiers are assigned to peers and resources.
- **Management of the Identifier Space:** explains which peers are responsible for a resource with a given key. It describes replication capabilities and how these are handled.
- **Graph Embedding:** shows how a peer selects its neighbors and builds its routing table.
- **The Routing Strategy:** describes how neighbors’ contacts are used to resolve the user query and perform overlay operations (e.g. publish, store, update).
- **The Maintenance Strategy:** describes how the failed contacts are detected and replaced and resources kept available.

In the following we analyze all design aspects of the four observed overlays (Chord, Kademlia, Gnutella 0.6, and Gia) [Nob09].

3.3.1 Choice of an Identifier Space

Identifiers of peers and resources and the closeness metric

The identifier space \mathcal{I} of an overlay must be large enough to avoid collisions and support a large number of participating peers. In all overlays, a resource is assigned to a value from the identifier space. In structured overlays, identifiers are also used as an overlay address for peers and to provide efficient, greedy routing. For the latter purpose, it

is required to define a *closeness metric* $d(x, y)$ between two keys $x, y \in \mathcal{I}$. In unstructured overlays, the closeness metric is not needed.

Chord uses a large integer identifier space from 0 to a maximal value 2^m (e.g. the range of SHA-1 values: $m = 160$). The distance function is the difference between x and y modulo 2^m : $d(x, y) = (y - x) \bmod 2^m$. The modulo operation ensures that the condition $\forall x, y : d(x, y) \geq 0$ for closeness metrics is fulfilled and allows the identifier space to be visualized as a *ring*. The distances between identifiers are bigger as an identifier point in the ring is further away clockwise. In Chord, if $d(x, y) = a$, then $d(y, x) = 2^m - a$. Thus, Chord's distance function is not *symmetric*.

Chord: identifier space from 0 to 2^m , with distance $d(x, y) = (y - x) \bmod 2^m$

Kademlia uses a similar identifier space to Chord, in terms of the size. Instead of taking the arithmetic difference as in Chord, a distance between two identifiers is compared by XOR operation, bitwise. The result is the distance value, treated as an integer (the larger 1-MSB in the result of $x \oplus y$ between two identifiers x and y , the greater the distance is between them). Because of the XOR operation, Kademlia's identifier space is *symmetric*, which means the distance from peer identifier x to peer identifier y is equal to the distance from y to x . The symmetry will result in symmetrical neighboring relations.

Kademlia: identifier space from 0 to 2^m , with $x \oplus y$ bitwise as closeness metric

Both **Chord** and **Kademlia**'s distance functions are *unidirectional*. If we pick a point, x , out of the identifier space and a distance, a , there will be exactly one y so that $d(x, y) = a$. This means, if two queries from a different initiator route to the same peer (more on this later in Section 3.3.5), both routing paths will finally meet during the query and take the final steps using the same hops. The authors of Kademlia exploit this property for caching values that are often looked up (will be detailed in Section 3.3.4).

In **unstructured** overlays (Gnutella 0.6¹ and Gia), there is no strict identification of the resources. The resources are addressed by their name or description. Finally, there is no *closeness metric*, because a peer that is close to a resource is unaware of its proximity (except when using QRP² or one-hop replication³ in Gia). As a peer identifier, usually only the transport layer address (IP address and port) of a peer is needed.

Gia and Gnutella 0.6: no particular identifier space for peers or resources

3.3.2 Mapping of Entities to the Identifier Space

As both peers and resources can be addressed by an identifier, we distinguish mapping of peers and of resources to the identifier space.

Peers

In all observed overlays, an identifier is assigned to each peer upon joining. In structured overlays the key assignment is usually performed

Upon joining, a peer obtains its identifier by hashing its IP address (in structured overlays)

¹ Gnutella 0.6 is hybrid, but the ultrapeers make an unstructured overlay

² QRP: Query Routing Protocol, a method for leaves to inform their ultrapeers about the stored resources to provide more efficient routing

³ One-hop replication tells all neighbors about the resources a peer shares for efficient routing at the last hop.

by hashing the peer's IP address. This aggravates certain Denial of Service ⁴ attacks, where attackers claim special identifiers. By comparing the sender's key to its IP address on each incoming message attempt, arbitrary claims of hashes can be excluded to a certain extent, as the sender's network layer address is likely to be reliable.

The mapping of peers to identifiers is *complete* in structured overlays, which means every peer is assigned to an identifier.

Resources

In structured DHT overlays, such as **Chord** and **Kademlia**, each resource stored in the network needs a unique overlay key. The key may be assigned freely, depending on the demands of the application layer, in many cases this is done by using any one of the following methods:

- a hash of a **keyword** that fits to the resource. For example, a user wants to make an IP telephony conversation and enters the user name he wants to call. The user name is hashed and looked up in the DHT. The DHT returns the actual IP address, the public key, the port number and the VoIP protocol type as the value, previously stored by the callee. Once this information is provided, the voice stream can be initiated and the user can start the conversation.
- the **hash of the resource** itself: This is usually done in file-sharing scenarios. The user retrieves the key either from an index of, from an URI from the web (e.g. the ed2k://... scheme of eDonkey) This method offers the advantage of the easy detection of corrupted resources, making the intentional placing of corrupted resources nearly impossible.
- a **random** bitstring: This is only used if the resource cannot be hashed (e.g. the resource is transient).

In structured overlays, each resource has a unique overlay key obtained by hashing

Note that in the case of keyword hashing or the use of random identifiers, advanced mechanisms to guard against attackers storing false resources in the network may be needed (e.g. digital signatures).

In **Gnutella 0.6** and **Gia**, the identifiers assigned to a resource are required to decide whether the resource matches a given query or not. Depending on the type of query, an identifier for a resource can be:

- a **set of keywords**: If a query for a particular keyword is made, the resource matches the query, when the query keyword matches one of the resource's keywords.
- the **hash of the resource or another UID of the resource**: If a query for a hash of the resource is made, the resource matches the query, when its hash/UID is equal to the queried hash/UID.

In unstructured overlays, the resource identifier is a set of keywords, UID, etc.

⁴ In this context, a Denial of Service (DoS) aims to make a certain resources or the entire peer-to-peer overlay network unavailable.

- **a complete structured description of the resource**, stored in a local database: A query for the resource is made SQL-like. A resource matches a query if it is part of the local query result. This is implemented in LimeWire, where a query for media can be made for artists, titles, genres, bitrates etc.

Unstructured overlay networks are not limited to DHT lookup operations which allows the implementation of more extensible and flexible queries. As we said, applications like Limewire support complex queries such as browsing for genres, dates and bitrates.

To browse for keywords in DHT networks *Inverted Indexes* are used. Here, the hash value of a keyword maps to a *list of resource keys* that match this keyword. A user can now lookup the hash value of a keyword in the network and examine the search results.

3.3.3 Management of the Identifier Space

In **Chord**, a peer is responsible for all resource keys between its own identifier and that of its predecessor (modulo m). Using the suggested replication mechanism, each peer manages the identifiers between its k -th successor's identifier and the identifier of its predecessor. k must be defined so that the simultaneous failure of k peers is very unlikely (in practice, $k = 20$).

Chord management:
all resource keys
between own
identifier and that of
its predecessor
(modulo m)

In **Kademlia**, each peer d is responsible for a resource key r , if $d(n, r) \leq e$, where e is the smallest distance so that at least k peers are responsible for the resource. The peer lookup operation of Kademlia lets the publisher of a resource easily determine this set of peers (as outlined in Section 3.3.5). Additionally, Kademlia supports the caching of resources. The peer on a lookup path, closest to the resource will store this resource (cache it) until a timeout expires. The timeout period is larger at close caching peers and decreases with the number of peers between the key and the peer itself, in order not to "overcache" the network.

*Kademlia
management:*
 k -closest peers
responsible for a
resource key r
($d(n, r) \leq e$)

In unstructured networks (**Gnutella 0.6 (Ultrapeers)** and **Gia**), each peer may maintain any identifier. Each peer holds a resource, either user- or application-layer-dependent (e.g. the shared folder of file-sharing clients), thus there is no *store* operation as in DHT networks.

*Gnutella 0.6 and Gia
management:* a peer
responsible for
resources it offers

3.3.4 Graph Embedding

Each overlay network can be visualized as a graph, consisting of a node for each peer, and an edge to each known *neighbor* of a peer. n is a *neighbor* of p , if p has the contact address of n (e.g. overlay address, the IP address and port). The network, thus the graph, is changing frequently, which means maintenance procedures for neighbor relations are required and are a crucial design aspect.

*Each peer stores
contacts (overlay, IP
add, port) of its
neighbors in a
routing table*

In **Chord**, the neighborhood relationship according to Aberer et al. [LAG⁺05] is $N(p) = \bigcup_{i=1}^m \{ft(p, i)\} \cup \{predecessor(p)\}$. $finger(p, i)$ is the peer's fingertable entry i . $finger(p, 1)$ is also called the *successor* of

*Chord routing table:
successors and fingers*

the peer p . The distance of each peer in the fingertable must fulfill the distance rule: $d(\text{finger}(p, i), p) \geq 2^{i-1}$. The finger table, as well as the latter rule, are very important to make the routing technique scalable, as will be discussed later in Section 3.4.1. In practice, where $m = 160$, the peer maintains a maximum of 161 connections, so that the *maximal out-degree* is fixed.

*Kademlia routing
table: a k-bucket for
each XOR distance
(160)*

In **Kademlia**, the neighborhood function is $N(p) = \bigcup_{i=0}^{m-1} \text{kbucket}(i)$, where each $\text{kbucket}(i)$ is a list of peers that the peer has recently received a message from. This means, Kademlia learns from received messages and does not actively look for peers to populate its routing table. The size of each k-bucket is restricted: $|\text{kbucket}(i)| < k$. Not every peer is allowed to be added to a k-bucket, it must first fulfill the condition $\forall j \in \text{kbucket}(i) (2^i \geq d(j, p) < 2^{i+1})$. As we can see, k-buckets with a small i , are peers close to p , where the buckets near m are more distant peers. The first k-buckets usually remain empty as the set of peer keys a k-bucket is able to manage is small at the lower indexes and is logarithmically growing with the bucket index. Due to the huge size of the key space, the probability that a neighbor's key belongs to a k-bucket with a small index is very slight. The upper boundary for the neighborhood (*maximal out-degree*) size of p is $\sum_{i=0}^{m-1} \max\{2^i, k\}$ which is 3131 ($155 \times 20 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 3131$) for $k = 20$. Thus, the neighborhood scales well, like in Chord.

Both **Chord** and **Kademlia** maintain their routing tables, ensuring *Local connectivity* as well as *Long-range connectivity*⁵, using rules for how the peers populate their routing table. This ensures the small *diameter* of the graph. Kademlia ensures more reliable connectivity by maintaining a *set* of peers for each distance step.

*Gnutella 0.6 routing
table: ultrapeers
connected with 32
other ultrapeers,
leaves connected with
3 to 5 ultrapeers*

In **Gnutella 0.6**, leaves maintain a connection to a maximum of 3 or 5 ultrapeers (Limewire). Ultrapeers commonly maintain connections up to 32 other ultrapeers. The ultrapeer concept makes Gnutella a hybrid overlay. In Gnutella, there is no *local-* and *long-range connectivity*, because there are no distances between identifiers. However, the probability that one of your neighbors has a similar neighbor table should be minimized, in order to avoid circles (redundant relays of query messages) and ensure a variety of query results. The *pong caching* mechanism ensures connections to equally distributed peers in the network.

*Gia routing table: a
peer is connected to
at least one
high-bandwidth peer,
the rest are random
chosen*

Gia behaves in a similar way to Gnutella ultrapeers. Gia uses a *topology adaptation* algorithm to ensure that peers with an inferior capability to handle many neighbors (e.g. a low-bandwidth internet connection) are assigned to peers with a good Internet connection, thus having many neighbors. This should lead to a small network utilization of low-bandwidth peers. Additionally, along with QRP, it should reduce the number of hops a query has to take before an appropriate number

⁵ The presence of local- as well as long-range connectivity contributes to efficient routing with logarithmic complexity of the hop count. While the first hops need long-range connectivity to reach peers in the wider area around the looked-up key, local connectivity ensures the last hops will eventually reach the very near peers

of results is returned. This decreases the graph diameter, thus the latency.

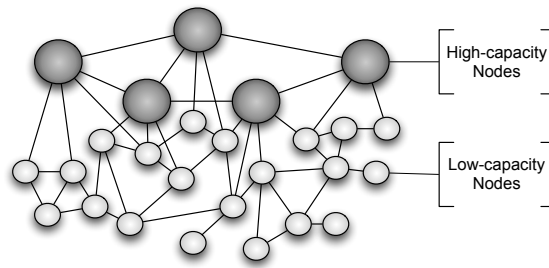


Figure 14: Gia topology adaptation: low-capacity peers keep connections to high-capacity peers and use them for routing

3.3.5 Routing Strategy

Routing strategy describes the usage of routing tables and neighbor connections, in order to route the overlay messages (e.g. query, updates, store, responses).

Chord

In Chord, a peer n looks up a key by choosing the closest peer in its finger-table (which is smaller than the key) as the next hop. If n is already the closest peer to the key, then n 's immediate successor is responsible for the resource, because it is also the immediate successor of the key. This routing strategy can either be implemented *iteratively* or *recursively*. In the first case, a peer that sends a lookup for a key sends a request to the next hop in its table and expects the second hop of the message as the response. The initiator then uses the received contacts and receives the response which includes matching or a closer peer. This produces many messages, as every query message hop starts and ends at the receiver. The advantage is that the initiator controls the query message and can detect its loss or the failure of a peer on the routing path immediately, which may improve the protocol's reliability. The *recursive* routing forwards the query from the initiator to the destination and then back to initiator, without contacting initiator following each hop. This implies much faster query response time and halves the number of introduced messages.

Chord routing strategy: Closest finger chosen, then it's successors (iterative or recursive)

Kademlia

Kademlia's routing mechanism determines the k closest neighbors to the looked-up key. Kademlia looks up iteratively and sends routing requests to multiple peers *concurrently*. Whenever a peer n wants to lookup a peer or a resource, it determines the α peers in its k -buckets that are the closest to the key (in practice, $\alpha = 3$). A peer n sends

Kademlia routing strategy: iterative, by forwarding a request to α closest contacts, who send their k -closest contacts to a key

a query message to these peers concurrently in order to receive their k neighbors closest to the key. The initiator n collects the received contacts and chooses the α closest peers to the key to send the query again. During this process, it refreshes its routing table and replaces the stale contacts with the fresh ones. This process is repeated, until the α closest peers return peers that were already queried. The retrieved k peers are the responsible for the value.

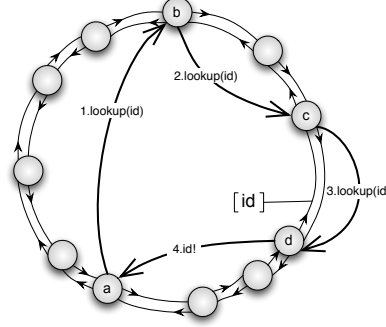


Figure 15: Example of the query resolution in Chord

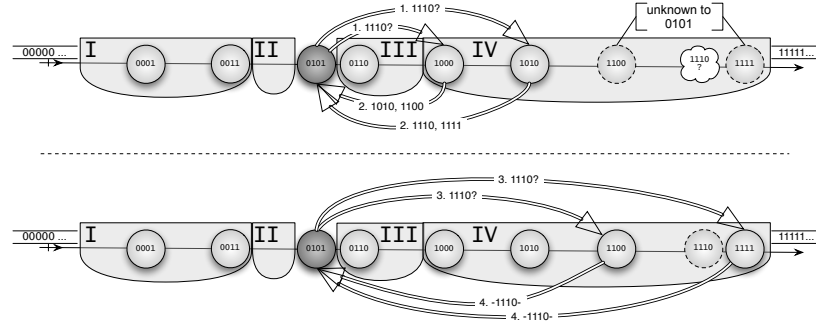


Figure 16: Example of the query resolution in Kademlia

Figures 15 and 16 show examples of resolving a lookup in Chord and Kademlia, respectively. In Chord, when a peer a starts a $lookup(id)$, it searches in its fingertable for the closest entry to a requested identifier id , in this case peer b (step 1). Peer a then sends a lookup message to the peer b , who also searches its fingertable for the entry closest to the id . It finds the contact address of a peer c and forwards the lookup query to it (step 2). Peer c searches in its finger table and recognizes that it itself is closer than any of its fingertable entries. Thus it selects its immediate successor, (d), as the result of the lookup query (step 3). Peer d sends the requested key with the identifier id to the peer a , who initiated the lookup (step 4).

When a Kademlia peer, with the identifier 0101 , starts a lookup for key 1110 , it first selects the $\alpha = 2$ peers from its k -buckets whose keys are closest to 1110 . It finds peers with the identifiers 1000 and 1010

from its fourth k-bucket. It then it sends a lookup message to both of them, at the same time (step 1). Both peers send back their k-closest entries, namely *1010*, *1100* and *1110*, *1111* (step 2). The initiator of the lookup then chooses the two closest peers out of all those it received (*1110* and *1111*) and sends them a lookup message (step 3). As a result, it acquire the requested key *1110*.

Gnutella 0.6

As already mentioned, routing in unstructured overlays is not greedy, meaning that the use of neighbor connections is not dependent of the query message. Gnutella (0.4 and 0.6) sends a query request to all or a set of its neighbors, which relay the request by sending it to their own neighbors and so on. The routing process itself scales poorly. If a peer receives a request, it browses for results in its local storage and returns a query hit containing all matching values to the query initiator. Gnutella 0.6's *dynamic querying* allows wide and shallow queries first, and conduct deeper searches on single peers (2-3 hops) if the shallow searches do not return a sufficient result count. In order to retrieve a variety of results, however, the algorithm should reach all regions of the network and not only peers in the immediate neighborhood. In Gnutella 0.6, *Pong Caching* ensures an equally distributed connectivity (see Section 3.3.4).

Gnutella 0.6 routing strategy: first to ultrapeers, then flooding limited with TTL

In order to solve the scalability issues of Gnutella 0.4's routing, Gnutella 0.6, introduces *leaves* and *ultrapeers*. A leaf supplies the network with a lot of resources, but has limited bandwidth and cannot withstand the flood of queries in the Gnutella overlay. Instead, a leaf connects to one or multiple ultrapeers, which have a lot of bandwidth to perform query routing. Thus, the ultrapeers act as a server for the leaves. A leaf sends Query Routing Table (QRT) to its ultrapeers, containing hashed information about the available resources on a leaf. When a leaf performs a query, it sends a query request to its ultrapeers for relaying the query. Whenever an ultrapeer receives a query message, it will look into the QRTs of its leaves and will forward the request only to the leaves that hold the resource.

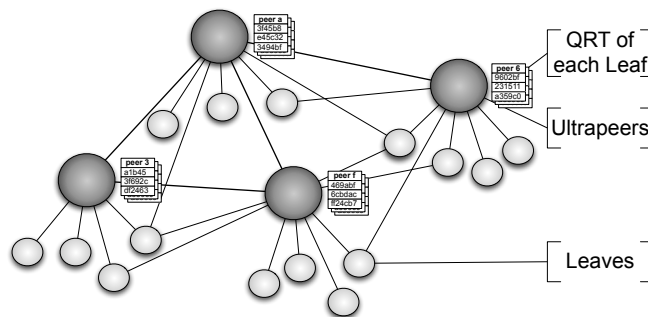


Figure 17: Gnutella 0.6 ultrapeers and leaves

Gia

Gia routing strategy: random walk, relies on high-bandwidth peers

As already mentioned, Gia improves of the scalability of the original Gnutella 0.4 in a purely decentralized fashion. Gia uses a random walk instead of flooding for routing. When a peer relays a query, it selects a peer that has a high network capacity. It is assumed that these peers store the resources with high possibility, or have a replica due to the *one-hop replication* of their neighbor's content. *One-hop replication* is similar to the QRP in Gnutella, but here, each peer sends information regarding its resources to its immediate neighbors. Therefore, a query is sent directly to the neighbor that has a resource matching the request.

3.3.6 Maintenance Strategy

Maintenance strategy: failure detection and failure recovery

An overlay needs to keep stored resources available, even if peers holding the resource go offline. Each peer in an overlay network needs to maintain its routing table, as peers may join and leave the network frequently, which would make the routing strategy inconsistent. Maintenance is a crucial design aspect for stability and robustness of the overlay. In order to maintain the contacts, failure of the peers must first be detected in a timely fashion and then an appropriate recovery mechanism must be applied. Therefore, in the following text we distinguish failure detection and failure recovery as the two maintenance steps.

Failure Detection

Failure of a peer is detected when the expected response from a peer is not received. Zhuang et al. [DGSK03] analyzed failure detection mechanisms in peer-to-peer overlay networks, and classified them into:

Active (ping, keep alive) and passive (piggy-backing with other messages) detection

- **Active** approach: The liveness information about a peer is sent periodically and actively. Depending on how the this information is received (and sent), we distinguish:
 - **Baseline**: where each peer independently makes a decision about the liveness of its neighbor. It sends or receives liveness information directly to its neighbors.
 - **Sharing**: where liveness information is shared between peers.
- **Passive** approach: The liveness information about a peer is “piggy-backed” with other protocol messages exchanged between two peers. This method is used rarely, as usually no continuous message between two peers can be assumed.

According to [DGSK03], a liveness information can be exchanged by *probing* or *gossiping*. Probing means sending a message “are you alive” to a neighbor, while gossiping is sending “I am alive” information to the neighbor. Either *positive* or *negative* liveness information can be sent (“a peer is alive” or “a peer is offline”). In order to detect the failure in a timely fashion, the frequency of periodical active failure detection

messages must be high enough. On the other hand, if exchanged too often, liveness information can increase protocol overhead, introducing additional overlay traffic. Careful parameterizing of failure detection mechanisms is crucial for maintenance and thereby for the robustness and stability of an overlay.

Failure Recovery

Aberer et al. [LAG⁺05] distinguish between proactive and reactive recovery mechanisms. *Proactive* failure recovery mechanisms can either be triggered by active failure detection mechanisms (**proactive recovery with active failure detection**), or done periodically, without previously detecting a failure (**periodical proactive recovery** without active failure detection). **Reactive** recovery mechanisms use passive failure detection and can be further classified into **correction on use**, **correction on failure**, and **correction on change** [LAG⁺05]. Figure 18 shows the difference between these failure recovery mechanisms.

Proactive (with active detection) and reactive (with passive detection) failure recovery

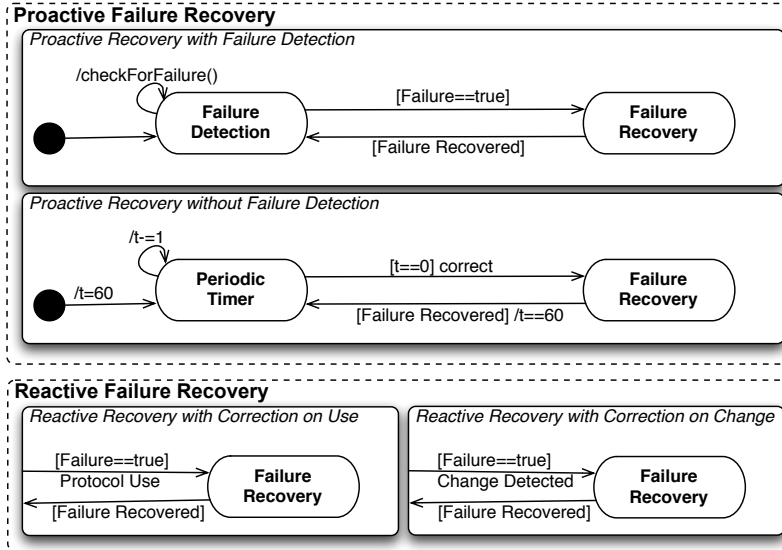


Figure 18: Maintenance strategies

We distinguish between *routing table maintenance* and *replication maintenance* in overlays that use replication, such as Chord and Kademlia. In the following, we present the maintenance of observed overlays separately.

Maintenance of contacts and replicas

Chord

Chord performs *finger* and *join stabilization procedure*, and replication maintenance. Finger stabilization procedure is a *proactive recovery with failure detection*:

Chord maintenance:
periodically checks
fingers, detection is
recovery at the same
time

- **Failure Detection:** Chord periodically updates its finger table entries by querying the finger identifier of each entry, using *active baseline* detection.
- **Recovery:** If a new manager of the queried finger identifier is returned as a result, failure is detected and corrected at the same time. If a successor fails, no response will be sent. Therefore, each peer additionally keeps in contact with all of its k immediate successors. An immediate successor will be removed after a timeout for response from a successor is exceeded.

When a peer n joins the Chord ring, it needs to have contact with at least one peer (bootstrap). A bootstrapping peer looks up the successor for n by querying the identifier of n in the network. A peer n sets the obtained contact as successor, who sets a predecessor to n . Now n is not aware of its predecessor, neither is n 's predecessor aware of n . Instead, n 's predecessor has a successor set on n 's successor. Thus, Chord has to contact a periodical stabilization procedure for this case too, not only upon failures.

Join stabilization procedure is also *proactive recovery with failure detection*:

Chord needs
periodical
stabilization because
of newly joined peers

- **Failure Detection:** Each peer m periodically queries its successor's predecessor x . A peer m detects a failure if x is greater than its own key. This is an *active baseline* approach, too.
- **Recovery:** A peer m sets x as its new successor and informs x about its new predecessor m .

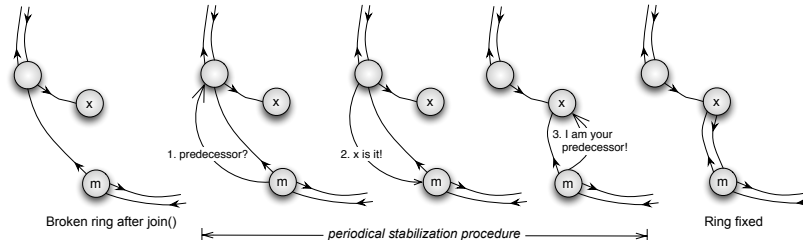


Figure 19: Stabilization process upon a peer join (fixing a broken ring)

In order to keep the resources stored in a Chord ring available, an application stores its information not only at the peer responsible for it, but also at its r following peers. If a peer n leaves, who is responsible for a resource, the next peer takes its over the responsibility. If all r peers have left, the resource would be lost, there must therefore be a mechanism to republish the keys. This can be done by the application layer of the initial resource published with a *periodic proactive recovery*, that republishes the resources periodically.

When a peer fails in
Chord, a successor
takes the
responsibility for its
keys

Kademlia

For its contacts, Kademlia uses both a *reactive recovery* and both *proactive recovery* types.

A peer n updates its bucket table on every communication attempt. Upon receiving a message, a peer n looks up the peer in the appropriate k -bucket and if it is not yet part of the routing table yet, it is appended at the end of the bucket. If the list is now overflowing (greater than k), the *first* peer in the bucket is pinged in order to check it is still alive. If the first peer does not work, it is dropped and the list is restored to its previous size. If the first peer works correctly, the *new* peer is dropped instead, and placed in a replacement cache. This helps to make the overlay robust against DoS attacks, where a new peer tries to flush a k -bucket of another peer by communicating with peers frequently using different identifiers.

Additionally, Kademlia uses a *proactive recovery with failure detection*:

- *Failure detection*: All bucket entries are pinged randomly, which is *active baseline* detection.
- *Recovery*: Stale contacts are simply discarded from the table and replaced with contacts from the replacement cache.

Kademlia also uses replication to improve the availability of resources. Values are stored at all k peers that are closest to the key of the resource. This is done via a *periodic proactive recovery* mechanism. Every hour, these peers republish the value, which stores the replicas on newly joined peers if some of the k peers have failed in a meantime. Every 24 hours, a content is republished by the original publisher. If republishing did not take place, the content will be dropped by the responsible peers (e.g. by using a timestamp of the original publishing time). This is usually done by the application layer and helps to keep the network clean from unused content.

Kademlia maintenance: a sender of a received message is added in routing table if it is not full or some of its entries are offline

Stored values in Kademlia are republished every hour by replica peers and every 24 hours by originator

Gnutella 0.6

Gnutella 0.6 uses *proactive recovery with failure detection* and *shares* liveness information. An ultrapeer m accepts or denies a contact to a peer n , depending on its contact list size limit, and replies with a list of ultrapeers known to it. The failure of a neighbor is detected and recovered in the following ways:

- An ultrapeer n periodically sends pings to its neighbors' ultrapeers, awaiting pong response within a timeout.
- The contact is removed from the routing table if the pong has failed once or multiple times, depending on the configuration.

Gnutella 0.6 maintenance: ultrapeers ping each other and share the liveness information

As discussed in Section 3.3.4 for low- and long-range connectivity and avoiding small cycles in the connection graph, a peer needs to maintain connection to distant peers in the network. As already mentioned, Gnutella 0.6 has numerous implementations, so there are multiple possibilities to achieve this [Gnu]:

- **Classic Multi-hop Ping**: In Gnutella 0.4, a ping is sent to all neighbors, which relay the ping to their neighbors and so on for

three or more hops. All of these hops respond with a pong that is routed backwards on the path of the ping message. This kind of procedure can produce a lot of traffic. It is still supported, but rarely used in current implementations of the Gnutella protocol.

- **Pong Limiting:** The number of pongs that are routed back to the source of the ping is limited, commonly to 10 pongs.
- **Pong Caching:** Every ultrapeer holds a local pong cache consisting of five other ultrapeers. The first one is one hop away, the second one is two hops away and so on. Whenever an ultrapeer is pinged, it replies with its local pong cache. Upon receiving a pong cache, a peer updates its local pong cache by storing the received entries into it, with a hop count increased by one. For example, a peer with the IP address 12 sends (My contact: (), 1 Hop: (IP: 36), 2 Hops: (IP 24)..) to n, n then stores the values (1 Hop: (IP:12), 2 Hops: (IP:36), 3 Hops: (IP 24)...) in its pong cache.
- **Pong Multiplexing:** Whenever a ping is received with a Time-to-live(TTL) greater than 1, the ping is relayed and the relaying peer waits for a responding pong. When a new ping is received from a different peer, before the pong of the last peer has received it, the ping is not relayed, but marked in the pong routing table (it is *multiplexed*). Upon receiving the pong from the first ping, the pong will be forwarded to both peers that made the ping (it is **demultiplexed**).
- **GWebCache:** This method is commonly used to initially fill the peers' routing table. A list of hosts can be placed on a webserver and accessed by other peers.

All of these methods, except GWebCache, distribute recently received liveness information to other neighbors, using *active sharing* detection.

Gnutella 0.6 maintain the QRP (Query Routing Protocol) between ultrapeers and leaves. Whenever a leaf connects to an ultrapeer or changes its documents, it submits its published content. This is a *reactive recovery on change*. When a peer disconnects or does not respond, the the corresponding hashed entries are removed from the ultrapeer's site, in order not to pollute the memory (its size is 8 KB). Failure recovery and detection of a leaf peer is *reactive on failure*.

Gnutella does not use replication mechanisms. Every peer serves its own files, so in a file-sharing application, popular content will often be downloaded and shared immediately after download. This results in indirect replication of popular content. However, rarely downloaded (thus rarely shared) content will not be available when a query message does not reach a peer who stores it.

Gia

Gia uses a *proactive recovery with failure detection*, like Gnutella 0.6, but does not necessarily send pong caches. The authors [CRB⁺03] do not

*Gnutella 0.6
ultrapeer maintain
QRP, a table with
hashes of content
stored at its leaves*

specify a method for how this can be done, they refer to the various strategies of Gnutella.

Gia uses a *flow control* mechanism to avoid the congestion of incoming queries. Therefore, a peer has to react to congestion by decreasing the token allocation rate for its neighbors, or increasing it if the bandwidth allows it. This is a *reactive recovery on usage*. Additionally, Gia introduces *one-hop replication*. Every peer sends information of its content to its immediate neighbors, to allow more efficient query resolution. Once the failure of a neighbor is detected, this information about the neighbors content is removed. Therefore, Gia uses *reactive recovery on failure* for its replicas.

*Gia maintenance:
pinging contacts*

*Gia uses replication
on a direct neighbor*

3.4 INFLUENCE OF DESIGN DECISIONS ON THE QUALITY

After all design decisions of the observed overlays are discussed, we investigate their influence on quality aspects in this section. We first analyze the possible effects of the described mechanisms on scalability, efficiency, fairness, robustness, and stability (Sections 3.4.1, 3.4.2, 3.4.3). We then run comparative simulative evaluations in order to draw the final conclusions about the quality influences of the design decisions. It must be noted that we outline the effects of the designed mechanisms but do not claim to have identified the exact reason (i.e. design decision) for certain behavior. In order to do so, a fine granular framework for all observed overlays must be developed to enable the exchange of overlay mechanisms without affecting other mechanisms. This is certainly the next step but not the focus or goal of this thesis. We used the evaluation methodology (metrics, workloads, evaluation tools) described in Chapter 5. Detailed simulation settings are given in Appendix B.

*How do these design
decisions influence
the quality?*

3.4.1 Scalability, Efficiency, and Validity

Scalability is the quantitative adaptability of the overlay to a changing number of participants or services in the overlay, while preserving the validity, efficiency, and load balance. We therefore observe scalability, efficiency, and validity at the same time, with a changing number of participating peers.

Scalability of the overlay operations is greatly affected by the routing strategy. If the number of introduced messages of e.g. queries, maintenance or publishing is directly proportional to the network size, a peer-to-peer overlay is not scalable. In **Chord** and **Kademlia**, the complexity of a lookup query is $O(\log(n))$, if the routing table is up-to-date. The maintenance complexity in Chord is $O(\log(n))$, as it performs query lookup for fingers as the failure detection mechanism and the number of fingers depends on the identifier size, not the network size. In spite of the fact that **Kademlia**, maintains a maximum of 3131 references ($k = 20, \text{idLength} = 160, 155 \times 20 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 3131$), it uses *reactive recovery on failure*, thus the maintenance complexity is $O(1)$.

*Scalability affected by
the routing. Chord
and Kademlia routing
complexity is
 $O(\log(n))$*

The maximum routing complexity of the original Gnutella 0.4, which uses random flooding, exponentially grows with a given Time-to-Live (TTL). However, if the TTL value is too low, too few peers are reached, resulting in bad retrievability. Increasing the TTL logarithmically with the network size, the same average retrievability can be achieved for routing complexity of $O(n)$.

Gnutella 0.6 introduces some mechanisms that greatly reduce the network load, but the routing complexity among ultrapeers remains the same. The most important improvement to Gnutella 0.6, regarding scalability, is the separation of the peer's behavior into ultrapeers and leaves. This excludes low-bandwidth peers from the high traffic in the network, while the ultrapeers have to also carry the traffic of their leaves. If we assume a query depth with a full network coverage, every query made will eventually have to reach every ultrapeer in the network.

We calculate the routing complexity on the following example. We assume a ratio of ultrapeers to leaves of $r = 20\%$, an ultrapeer-to-ultrapeer degree of $d_{up} = 32$, and a leaf-to-ultrapeer degree of $d_{leaf} = 3$. A "perfect" graph (this means there are no circles smaller than 4) is therefore assumed.

We define the average leaves that are connected to an ultrapeer in the following way: $avgLvs = d_{leaf} \cdot (\frac{1}{r} - 1) = 12$. Here, $\frac{1}{r} - 1$ is the number of leaves per ultrapeer. Since a leaf is connected to three ultrapeers, this is additionally multiplied with d_{leaf} .

A query that reaches an ultrapeer, reaches its $avgLvs = 12$ connected leaves, too. A query depth of $d = 3$ reaches $32 + 32^2 + 32^3 = 33824$ ultrapeers, thus approximately 439 thousand peers, a query depth of $d = 2$ reaches a maximum of $32 + 32^2 = 1056$ ultrapeers, thus 12.6 thousand peers altogether. $d = 3$ reaches many peers in the network, but introduces an enormous amount of bandwidth. If we assume a query has a size of 20 bytes (which is very small), the traffic introduced by a single query is already 700KB. A query depth $d = 2$ produces an overhead of 21 KB which is acceptable, but it reaches fewer peers, resulting in poor retrievability. We can, therefore, expect poor scalability of Gnutella 0.6 for more than 500 thousand peers.

Gia deals with the scalability issues of original Gnutella 0.4 with the combination of *topology adaptation*, *biased random walk* and *one-hop replication*. First, Gia's *topology adaptation* brings low-degree peers into the proximity of high-degree ones. When a query is started, it uses a *biased random walk* to forward the query to the peer in its routing table with the highest available degree. *One-hop replication* decreases a hop count by replicating the content at the immediate neighbors.

In the following we present the simulation results on the scalability and efficiency of the observed overlays. Performance is presented by hit rate and query response times and costs with the stale message ratio and number of hops needed to resolve a query.

Figure 20 shows the comparison of hit rates in all four observed overlays. We can see that Chord has a significantly lower hit rate, in spite of reduced churn rate in contrast to other overlays, due to its strict neighborhood selection. Unlike Kademlia, Gnutella 0.6, and Gia, Chord

Routing complexity of Gnutella 0.6 depends on query depth

Gnutella 0.6 produces 700KB of traffic in a single query

Gia introduces biased random walk and one-hop replication to deal with the scalability issues

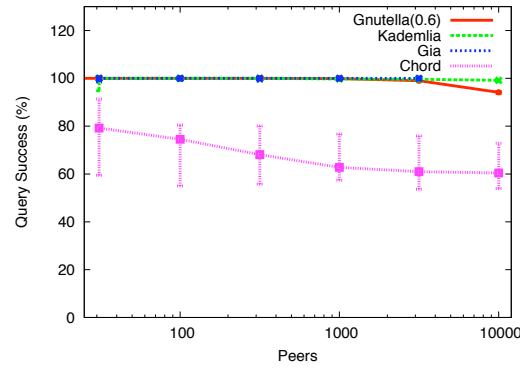


Figure 20: Hit rate with the increasing network size

is not able to resolve a query when contacts in the routing table are stale. Furthermore, we can see a decrease in the hit rate for Gnutella 0.6 in a network with 10 000 peers. The decrease in performance is occurring earlier than we assumed, in spite of the fact that the TTL is set to 2 in the simulations. Gia and Kademia have a hit rate of 100%, in spite of the presence of a churn and large network size.

*Gia and Kademia
very scalable
regarding hit rate*

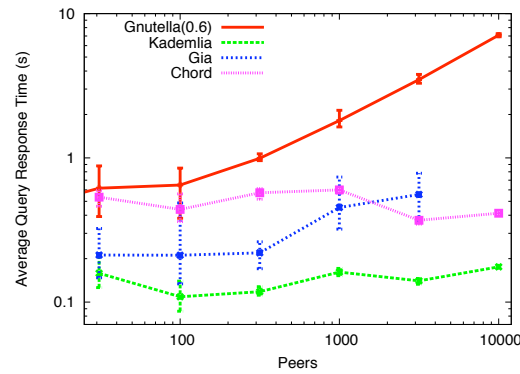


Figure 21: The average query response time. Only succeeded queries are included

Figure 21 shows the average query response times of all four overlays. With an increasing network size, **Gnutella 0.6** needs significantly more time to resolve the queries than the other overlays do. The main cause for this is a timeout for broadcasting a query among ultrapeers. A query is sent to a neighbor and if no result is retrieved before the timeout of one second is exceeded, a query is forwarded to the next neighbor and so on. In the worst case, the query response time is equal to the number of neighbors of an ultrapeer (in our simulations it is 32, meaning a worst case query response time of 32 seconds). **Chord** needs approximately 500 ms to resolve a query without a significant increase with the growing network size. The query response time of **Gia** is around 200 ms. Only a small number of hops is needed before a high-degree peer is reached, which probably stores the resource via

*Gnutella 0.6 needs
over one second,
Chord 500 ms, Gia
200ms, and Kademia
150 ms to resolve a
query*

a one-hop replication. In larger network sizes, however, the duration of query resolution begins to increase. Many high-degree peers have to be contacted until one with a replicated resource is found. Due to its concurrent lookups and large routing tables, **Kademlia** needs on average 150ms until a query result can be returned.

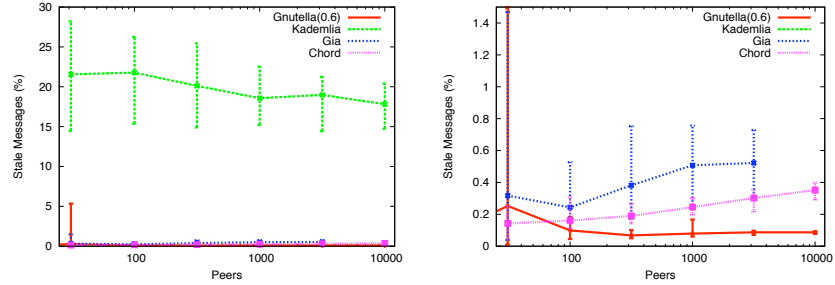


Figure 22: Share of stale messages

Figure 22a shows the share of messages sent to the dead contacts. As the maintenance of the routing table of a Kademlia peer is *reactive*, there is a large amount of stale contacts in the routing table of a peer. This results in around 20% of sent stale messages in Kademlia, which is around 10 times more than for the other overlays. Figure 22b, shows the share of stale messages for Chord, Gia, and Gnutella 0.6. Due to proactive recovery with failure detection, Chord has a significantly lower stale contacts rate than Kademlia, at only 0.2%. Gnutella 0.6 has the lowest stale contacts ratio of all of the overlays (on average 0.1%). It is the result of the short ping interval, which is smaller than in Gia because the failure detection is baseline instead of sharing, and does not expand additional traffic over the neighboring peers.

The number of hops needed to resolve a query in the observed overlays is presented in Figure 23. Our concerns about the query routing complexity in Gnutella 0.6 and Gia proved to be correct, as the number of hops in both overlays increases linearly with the network size. The number of hops in Kademlia and Chord stays almost constant,

Kademlia has over 20% of stale messages, while other overlays under 0.2%

In Kademlia and Chord, hop counts remain almost constant with the growing network size, unlike Gia and Gnutella where it linearly grows

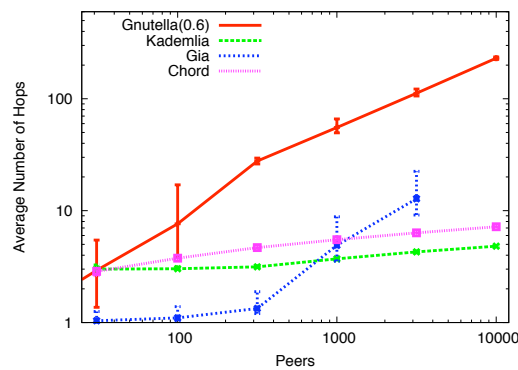


Figure 23: Average number of hops needed to resolve a query

with logarithmic increase with the network size (not more than 4 hops).

Summarized, we saw the vulnerability of Chord to churn, resulting in rate of almost 40% of unsuccessfully solved lookup queries. The hit rate in Gnutella 0.6 is decreased by 2% when the network size reaches 10 000 peers. Gia and Kademlia have 100% hit rate independent of the network size. The average response time of the query is the worst in Gnutella 0.6 (from 700 ms to 7s) and linearly grows with the network size, like Gia. However, a query response time in a network of 10 000 peers is up to 500 ms in the case of Gia. Kademlia and Chord have an almost constant response time which is 500 ms in the case of Chord and 150 ms for Kademlia. Reactive failure recovery of Kademlia results in over 20% of stale messages, which is around 10 times more than in the other overlays. The average hop count is drastically larger in the case of Gnutella 0.6 than for the structured overlays and it scales linearly with the growing network size.

We saw a big difference in the scalability of unstructured and structured overlays in terms of the number of hops and the query response times. The big influence on both scalability and efficiency was taking the closeness to the searched resource into account when routing (in next-hop selection). Not only Chord and Kademlia do this by using key-based routing, but, indirectly, Gia as well. The query in Gia is routed to the high-degree peer, which most probably stores the searched resource via one-hop replication. In spite of introducing ultrapeers and leaves in the network structure and routing, Gnutella 0.6 inherited poor efficiency and scalability from the original Gnutella as the majority of routing is done with random flooding. Additionally, timeouts on each hop in recursive routing (Gnutella 0.6) increase query response time. Instead, either iterative routing or broadcasting to the small subset of peers should be used while routing (as in Kademlia, with α parallel lookup queries). A rigid neighbor selection and routing, like in Chord, can lead to unstable behavior when peers join or fail.

The big influence on scalability and efficiency is using closeness metric in next-hop selection

3.4.2 Fairness

Fairness in peer-to-peer overlays means the uniform distribution of the costs for overlay operations over the peers. The load of a peer should be proportional to its individual capacity.

In **Chord**, only one peer is responsible for a key. Therefore, a deterministic routing always leads to this peer when a user starts a lookup for the key. Very popular documents can cause an overload of the peers responsible for them.

Kademlia introduces a caching mechanism to face the balancing problem that would appear in the case of popular files. When a query is finished, the corresponding key-value pair is additionally stored at the nearest peer in the query table of the peer that does not have the key. Kademlia's caching mechanism answers popular queries before it reaches the k closest peers responsible for it, thus removing load from them.

Fairness, when a file is very popular, may be critical in Chord, while Kademlia solves it with caching

Gia uses a token-based flow-control mechanism to achieve fairness

In **Gnutella 0.6**, leaves handle only small amount of traffic for keeping their connections to ultrapeers. In contrast, ultrapeers have to deal with high-bandwidth traffic, depending on the query depth. Due to the fixed in- and out-degree, an ultrapeer will receive query requests only from peers directly connected to it. The Gnutella 0.6 ultrapeers form an unstructured network, which has the advantage that popular documents are inherently distributed throughout the network.

Gia directs queries to high-degree peers, thus they may tend to become overloaded. These peers have to handle and forward all of their neighbor's queries. The token-based flow control mechanism directs a query to the peer in the next degree down, if the tokens for forwarding are depleted. The tokens are assigned according e.g. to a user-defined bandwidth. The flow control mechanism introduces an upper boundary for a peer's traffic, but it may still leave peers with middle and low capacities without any work, except relaying their own queries. Gia's flow control mechanism directs queries to lower-degree peers, if high-degree ones are not available. This may have an impact on the coverage of the query.

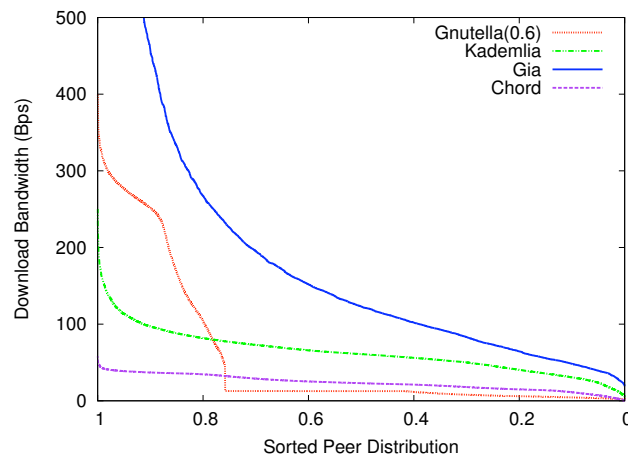


Figure 24: Distribution of the consumed download bandwidth per peer

Gia and Gnutella take heterogeneity of peers into account

Figure 24 shows the distribution of download bandwidth consumption. In Gia, around 10% of the peers have to carry more than 500 Bps, while 60% consume less than 200 Bps. This shows the support for heterogeneity of available bandwidth of the participating peers. More than 75% of the peers in Gnutella 0.6 consume only very little bandwidth (the fraction of leaves). Then (from right to left) there are around 10% of peers with bandwidth consumption of around 50 Bps and then a continuous increase up to 400 bps (the fraction of ultrapeers). This unequal load among ultrapeers may result from joining phase. When a leaf joins the network, it contacts the first available ultrapeer that has free slots (e.g. via web cache). In Kademia, only 2% of the peers consume 200 bps or more, which is around 2,5 times more than the average bandwidth consumption of a peer. Chord proved to be well load balanced, with a very small fraction (less than 0.1%) of peers

In Chord and Kademia, the most loaded peer is twice as loaded as the median loaded peer

consuming around 50 Bps, which is around twice that of an average loaded peer.

We can conclude that using Kademlia or Chord can result in an overload of low capacity peers, when peers' capacity is heterogeneous. Their routing strategy and graph embedding does not take peer capacity into account. However, connecting too many peers to a high-capacity peer can, again, result in uneven distribution among high-capacity peers.

3.4.3 Robustness and Stability

Robustness is the persistence of a peer-to-peer overlay when crucial parts of a system fail, while stability observes the system persistence under system perturbations such as intensive or frequent overlay operations (e.g. queries, leaving, joining).

The main design decision for achieving robustness and stability is timely maintenance.

As we have already seen, **Chord** has severe stability problems under churn, due to its strict neighbor selection and routing topology. If new failures occur during the stabilization process of fixing a broken ring, it can result in multiple independent rings.

Kademlia has a much larger routing table than Chord – 160 k-buckets containing k contacts each, resulting in a maximum of 3131 contacts, instead of only a single finger per entry, resulting in 171 entries in routing table in Chord. Additionally, a Kademlia peer sends query request to α peers in parallel. It is, therefore, very unlikely that routing will fail due to stale contacts. In order not to produce enormous overhead by periodical failure detection messages, Kademlia uses *reactive failure recovery*. As we saw, it results in more than 20% of messages sent to the stale contacts.

In **Gnutella 0.6**, both leaves and ultrapeers keep the connections to multiple ultrapeers. When a neighbor fails, it is simply replaced by other contacts, which are kept fresh with pong caches. In **Gia**, the mechanism to ensure stability is nearly the same. Peers stay connected to many neighbors and add a new contact if an established connection fails.

Figures 25 and 25 show the hit rate and share of stale messages when at time point t_0 , 50% of the peers simultaneously fail. Because of the extreme instability of Chord, only 10% of its peers fail at that moment.

Kademlia shows the best stability of all the overlays. Its query success stays at 100%, due to its large routing table and the iterative query resolution. In Gia, around 8% of the queries fail. Gnutella has a higher rate of failing queries. A leaf sends a query to an ultrapeer which handles it. If the ultrapeer fails during this query, the result will not be returned. Thus, for approximately 40 seconds, 30% of the queries fail. Chord shows the worst performance decrease and does not recover, even if only 10% of the peers fail. Around 50% of the queries fail after t_0 .

*Chord unstable,
Kademlia very robust
even when 50% peers
fail*

*Gia and Gnutella 0.6
have no robustness or
stability issues*

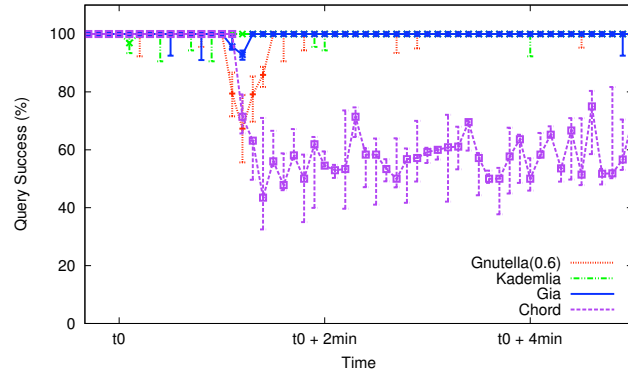


Figure 25: Decrease of query success when 50% of the peers simultaneously fail at t_0 (except Chord where only 10% peers fail)

In Figure 26, we see the ratio of stale messages sent after t_0 . All overlays experience a peak of stale messages ratio which drops back to zero after 20 seconds. Only Kademia continues sending messages to stale contacts, rather than removing them from their routing table.

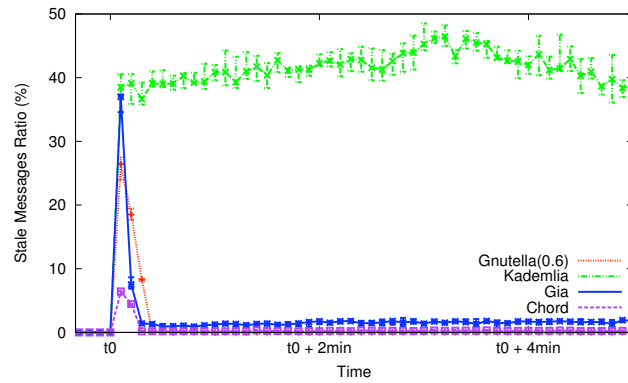


Figure 26: Share of stale messages when 50% of the peers simultaneously fail at t_0 (except Chord where only 10% peers fail)

Summarized, we can see that large routing tables with low and long-range connections and parallel, iterative routing, improve robustness significantly. On the other hand, rigid neighbor selection and routing strategy makes the overlay vulnerable to changes in participating nodes, especially failures.

3.5 LESSONS LEARNED

We observed four search overlays of different types which are well established in research and deployed in practical peer-to-peer applications. They are designed mostly for file-sharing applications or distributed storage. In spite of their different purposes (not for fully retrievable search, but one lookups), we can learn from their designs and de-

rive some conclusions regarding our main goal and the corresponding requirements.

We show the significance of the relation between the management of identifier space and graph embedding (property of structured overlays) which enables greedy routing and increases validity and efficiency of query resolution. However, the very strict bond between them together with rigid neighbor selection and routing strategy makes the overlay unstable and vulnerable to failures of the peers. Keeping local and long-range contacts plays a big role in efficiency, validity, and robustness. In order to achieve greedy and efficient routing, a peer identifier space must be chosen to relate to the user queries. Furthermore, a notion of closeness to the searched resource while routing and choosing the next hop must exist. To improve load balancing corresponding to the peer capacity, an overlay design must take heterogeneity of the peers into account. Timely failure recovery reduces the number of messages sent to unavailable peers and improves robustness.

*Design decisions
crucial for quality of
an overlay*

For our goal, i.e. building a peer-to-peer overlay for fully retrievable location-based search, only structured indexing and routing would guarantee completeness of the results. That implies peer location information in the peer identifier space to support greedy routing to resolve location-based searches. Both local and long-range contacts must be included in the routing table. As location-based search is two-dimensional, the management of the identifier space must be two dimensional, too rather than of one-dimensional as it is case in DHTs (ranges of the hash values).

*A foundation for
building our overlay*

3.6 SUMMARY

This chapter described an important step in engineering a peer-to-peer overlay networks – acquiring knowledge about existing design decisions from relevant solutions. We first presented the taxonomy of existing overlays regarding their purpose, structure, and query types. We analyzed the design decisions of structured (Chord and Kademlia), unstructured (Gia), and hybrid (Gnutella 0.6) peer-to-peer overlays which are well established in research and deployed in practical peer-to-peer applications. We observed six classes of design decisions, according to the reference architecture model of peer-to-peer overlays from Aberer et al [LAG⁺05]. In order to see the influence of these design decisions on the quality aspects, we analyzed the possible effects and performed comparative simulation evaluations.

*Taxonomy of overlays
presented, Chord,
Kademlia, Gia, and
Gnutella 0.6 chosen
for further analysis*

In our evaluations of *scalability and efficiency*, we saw the big difference between unstructured and structured overlays, in terms of the number of hops and query response times. Hit rate in Gnutella 0.6 decreased by 2% when the network size reached 10 000 peers. Gia and Kademlia had 100% hit rate independently of the network size. Average response time of the query was the worst in Gnutella 0.6 (from 700 ms to 7 s) and linearly grew with the network size, like Gia. However, a query response time in network of 10 000 peers is up to 500 ms in the case of Gia. Kademlia and Chord had almost constant response

*Summary of
comparative
evaluations of
observed overlays*

times which were 500 ms in the case of Chord and 150 ms for Kademlia. The reactive failure recovery of Kademlia results in over 20% of stale messages, which was around 10 times more than in the other overlays. The average hop count is drastically bigger in the case of Gnutella 0.6 than for structured overlays and it scaled linearly with the growing network size.

Regarding *fairness*, in Gia around 10% of the peers had to carry more than 500 Bps, while 60% consumed less than 200 Bps. More than 75% of the peers in Gnutella 0.6 consumed only a little bandwidth. There are around 10% of peers with bandwidth consumption of around 50 Bps and then continuous increase up to 400 bps. This unequal load among ultrapeers may result from the joining phase. When a leaf joins the network, it contacts the first available ultrapeers that have free slots (e.g. via web cache). In Kademlia, only 2% of the peers consume 200 bps or more, which is around 2,5 times more than the average bandwidth consumption of a peer. Chord proved to have good load balancing, with very small fraction (less than 0.1%) of peers consuming around 50 bps, which is around twice that of an average loaded peer.

Kademlia shows the best *robustness* of all the overlays. Its query success stays at 100%, due to its large routing table, and the iterative query resolution. In Gia, around 8% of the queries fail. Gnutella has a higher rate of failing queries. A leaf sends a query to an ultrapeer which handles it. If the ultrapeer fails during this query, the result will not be returned. Thus, for approximately 40 seconds, 30% of the queries fail. Chord shows the worst performance decrease of 50% and does not recover, even if only 10% of the peers fail.

*The main conclusions
of this chapter*

We concluded that greedy and efficient routing increases the validity and efficiency of query resolution. In order to achieve greedy and efficient routing, a peer identifier space must be chosen to relate to the user queries. Furthermore, a notion of closeness to the searched resource while routing and choosing the next hop must exist. However, very rigid neighbor selection and routing strategy make the overlay unstable and vulnerable to the failures of the peers. Keeping local and long-range contacts, plays an important role in efficiency, validity, and robustness. To improve load balancing, corresponding to the peer capacity, an overlay design must take the heterogeneity of the peers into account. Timely failure recovery reduces the number of messages sent to unavailable peers and improves robustness.

Based on these findings, the basis for the suitable design choices to realize our goal is given: it is a structured overlay with the identifier space containing the information about offered resources and a notion of closeness to the searched resource while routing and choosing the next hop must exist. As location-based search is two-dimensional, management of the identifier space must also be two dimensional, rather than one-dimensional, as is case in DHTs (ranges of the hash values).

The main goal of this thesis is engineering a peer-to-peer overlay network that enables fully retrievable location based searches. It meets the requirements stated in Section 1.2 and the design decisions are based on the comparative analysis and evaluation of the existing peer-to-peer overlays, presented in the previous chapter. This overlay network, its protocol and used mechanisms are in further text referred as Globase.KOM which stands for **G**eographical **L**ocation **B**ased **S**earch.

After all assumptions for the design are described, design decisions are presented according to the reference architecture of peer-to-peer overlays given by Aberer et al. [AH02]. Details of certain design aspects are discussed in separate sections.

4.1 ASSUMPTIONS

The design of Globase.KOM assumes geolocation of the peers, an appropriate map projection and defines resources and their descriptions, while taking the heterogeneity and mobility of peers into consideration.

4.1.1 Geolocation

We assume that each peer is aware of its geographical location. The assessment the location can be performed in any of the following ways:

- *Using GPS (global positioning system) receivers*, integrated into many mobile phones, PDAs, vehicles and portable navigation systems. As the chipset prices have dropped, GPS receivers are expected to find many new applications and to be integrated into standard laptops and other end-systems. Revenue from the sale of GPS chipsets is expected to grow from the current \$729 million to nearly \$1.3 billion in 2013 [Dato8]. The precision of geolocation using GPS devices is about 20 meters outdoors. They are, however, ineffective indoors.
- *Mapping from an IP address* is a non-invasive yet imprecise geolocation. It uses different mapping techniques, such as querying the Whois server [WHO85], exploiting the user-entered location data on web sites, extracting the relevant information in the DNS name of an Internet host, or the delay-based technique relying on the empirical measurements of network delay. The description and analysis of available techniques can be found in [PS01]. There is a lot of geolocation software available, such as MaxMind [Max09], IP2Location [TM09], or IP2Geonet [IP209]. The precision is unfortunately very low. Despite it having [Max09] 99.8% accuracy

This chapter presents a novel peer-to-peer overlay for fully retrievable location-based search

How do peers assess their geographical location? Both the process of getting this information and the information itself are referred as geolocation.

at a country level, the median error of this geolocation varies from 28 kilometers to several hundred kilometers. According to [Max09] the possibility of an incorrect geolocation for more than 40 kilometers is between 3% and 45%, depending on the country.

- *Using a Wi-Fi connection location* which uses radio beacons such as 802.11 access points, GSM cell phone towers, and fixed Bluetooth devices for geolocation [Pla], [LCC⁺05b]. The location of these beacons, identified by unique or semi-unique ID, is used for the estimation of a clients' location. Unlike geolocation through GPS devices, this technique works just as well indoors as outdoors. Accuracy is very high - median error varies from 13.5 metres to 31.3 metres.
- *Geocoding* transforms manually entered postal addresses to geolocation based on online mapping tools, such as [Inco9], [GMA09a], [Geo09], or [GMA09b]. The accuracy is high (up to hundreds of meters) for industrialized countries like USA, Canada, Japan and most EU countries. However, for developing countries, precision is on country level.

*Representation of
geolocation and map
projection used*

The description of geolocation is in the form of *longitude* and *latitude* values in DMS format, which is standard in geographic information systems (GIS) and Global Positioning Systems (GPS) (e.g. 44°49'14"N 20°27'44"E for Belgrade). Further details on geographical coordinate system and Plate Carée map projection which is used can be found in Appendix C.

4.1.2 Heterogeneity of Peers

*Heterogeneity of
peers in terms of
capacity, network
connection and online
behavior is both
supported and
exploited*

Resources and capabilities of Internet participants vary greatly - some may be mobile end-devices with little storage capacity, processing power, and poor network connection, while others might be powerful desktop machines or even server-like machines with large RAM and good network connectivity. Furthermore, the online behavior of users (particularly in peer-to-peer systems) differs. Some may be online for very long durations, while others are not. To learn about the online behavior, a mechanism much like the burn-in optimization of [Dar05] can be used. Strategies for superpeer selections are discussed in [ZCWQ08].

The heterogeneity of the participating peers is both supported and exploited. More powerful peers with good network connectivity, which are publicly reachable, and tend to stay online for long periods of time get more significant role in the overlay network. They have additional responsibilities in indexing, replication and caching (see 4.3). The "regular" (non-super) peers in the network simply offer and consume services without having additional responsibilities.

4.1.3 Mobility of Peers

In Section 1.2.1, we say that support for mobile users must be considered in some extent. It also says that we assume users are searching for static objects - gas stations, restaurants, emergency stations etc. Therefore, all providers of the services are static while their users can be mobile. In the design of Globase.KOM, mobile peers are treated just as clients, unlike pure peer-to-peer systems where each participant both provides and consumes services.

Mobile peers are assumed not to be providers of services

4.1.4 Description of Resources

We assume that peers provide information about the services related to their geolocation. The services can be chat, live streaming, reservations, online shopping etc. The resources can be described by their geolocation and optional metadata description (e.g. hotel, hospital, restaurant). An example of a query, an area search in Globase.KOM is 'find all peers in the radius of 50 km around Darmstadt'

Resources in Globase.KOM are described by geolocation of the peers and optional metadata

4.2 A BRIEF OVERVIEW OF GLOBASE.KOM OVERLAY NETWORK

Globase.KOM is a superpeer-based overlay forming a tree enhanced with interconnections. Superpeers are chosen from publicly reachable, static peers with more capacity, spare bandwidth, and good network connectivity. The world projection is divided in rectangular, not overlapping zones, as presented in Figure 27. Each zone is assigned to a superpeer that is located inside of the zone and keeps overlay/underlay contact addresses to all peers in that zone. Superpeers form the tree where superpeer A is called the parent of superpeer B when B's zone is inside A's zone.

Globase.KOM is a hybrid structured overlay, with 2D index

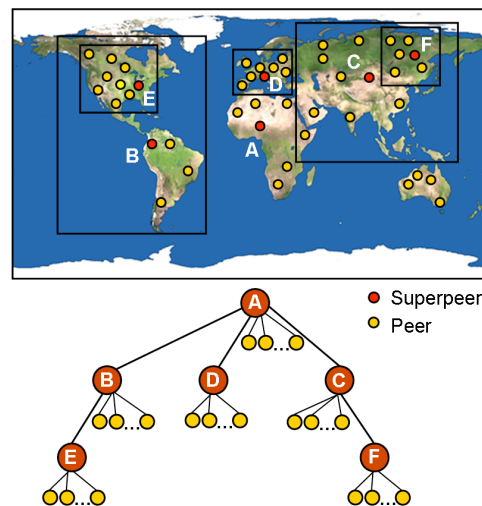


Figure 27: Basic structure of the Globase.KOM overlay

Beside parent and children connections, each superpeer maintains interconnections to the superpeers in the higher levels of the same branch or to the superpeers from the different branches. The inner zones are formed once a number of peers inside exceed a load threshold. A newly formed zone is then assigned to one peer in that area, that fulfills the requirements for becoming a superpeer. PeerID has information about the zone of responsibility and the geolocation of a peer. When a peer in the zone of superpeer B initiates an *area search*, it sends a query message to superpeer B. In the overlay and zone structure from Figure 27, if the zone does not intersect the zone superpeer B is responsible for, the query message will be forwarded to the superpeer A. This process continues on each superpeer in the path. Eventually, superpeers A, C, and D will reply with the list of the matching results. All details will be further described in the following sections.

4.3 DESIGN DECISIONS

Design decisions are explained according to the reference architecture of p2p overlays [LAG⁺05]

Globase.KOM is a structured hybrid overlay. Peers \mathcal{P} are connected to the responsible superpeers $\mathcal{P}_s \in \mathcal{P}$ which makes the overlay hybrid. Superpeers are connected in a structured overlay and there is a relation between resources and the overlay topology. The design decisions of Globase.KOM are presented using reference model [LAG⁺05] described in Section 3. In the following text both regular peers and superpeers are referred as 'peers'.

4.3.1 Choice of an Identifier Space

As previously discussed in 4.1.4, peers provide services related to their location and information about them. Resources are addressed by their location and metadata description from the predefined set of categories. This implies that both peers and resources can be addressed with the identifier containing their location and metadata description. However, for more efficient communication (see 4.3.3), the peer identifier contains additional information about the management of the identifier space. The identifier space of peers and resources in Globase.KOM is not identical but nonetheless related.

Peers

The identifier of a peer contains geolocation, zone of responsibility, and random part

The identifier space $\mathcal{I}_{\mathcal{P}}$ of peers in Globase.KOM is from 0 to 2^{184} . It is the concatenation of three natural numbers *loc*, *zone* and *rand* (like presented on Figure 28) so that

$$f: \mathbb{N}^3 \rightarrow \mathbb{N}$$

$$f(i_{\text{loc}}, i_{\text{zone}}, \text{rand}) : 2^{\uparrow \log(\text{zone}) + \uparrow \log(\text{rand})} \cdot i_{\text{loc}} + 2^{\uparrow \log(\text{rand})} \cdot i_{\text{zone}} + \text{rand}$$

where:

- i_{loc} is a representation of geolocation *loc* (represented by $\uparrow \log(\text{loc}) = 58$ Bits),

- i_{zone} , where zone is the zone a superpeer is responsible for, is a representation of it (see 4.3.3). This representation contains its geolocation of bottom left and top right point of a rectangular zone. In the case of regular peers, this field is filled with zeros. In that way we can identify the type of the peer (a regular peer or a superpeer) just by its identifier. It uses $\uparrow \log(zone) = 116$ Bits for its representation.
- rand is a random part to support the existence of more than one thousand peers in the same location. The number of Bits for the representation of the random part rand is $\uparrow \log(rand) = 10$.

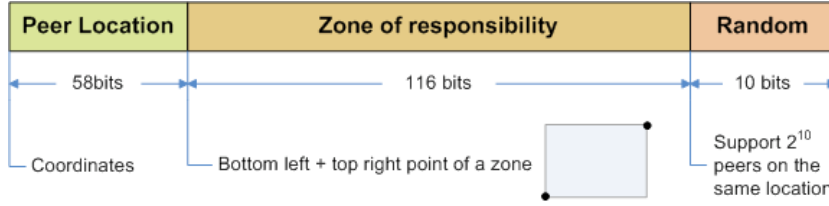


Figure 28: Structure of Peer Identifier Space

The set of superpeers \mathcal{P}_s is defined as $\mathcal{P}_s = \{X \in \mathcal{P} \mid i_{zone_x} \neq 0\}$.

We define $\subseteq: \mathcal{P} \times \mathcal{P} \rightarrow \{0, 1\}$ so that:

- $X \subset Y = 1$ for $X, Y \in \mathcal{P}_s$: $zone_x \subseteq zone_y$ or $X \notin \mathcal{P}_s, Y \in \mathcal{P}_s$: $loc_x \subset zone_y$,
- $X \subset Y = 0$ otherwise

Further, we define $\cap: \mathcal{P} \times \mathcal{P} \rightarrow \mathcal{P}$ so that:

- $\forall X, Y, Z \in \mathcal{P}$: $X \cap Y = Z$ if $zone_x \cap zone_y = zone_z$,
- $\forall X, Y \in \mathcal{P}$: $X \cap Y = \emptyset$ if $\neg(X \subset Y \wedge Y \subset X)$.

A closeness metric $d_P: \mathcal{P} \times \mathcal{P} \rightarrow \mathbb{N}$ for the identifiers x and y of the corresponding peers X and Y is defined as

$$d_P(x, y) = \begin{cases} 0, & \text{if } x = y \vee loc_x = loc_y; \\ 1, & \text{if } X \subset Y \wedge (\neg \exists Z \in \mathcal{P}_s): X \subset Z \subset Y, \\ & \text{if } Y \subset X \wedge (\neg \exists Z \in \mathcal{P}_s): Y \subset Z \subset X; \\ 2, & \text{if } X \subset Y \wedge \exists Z \in \mathcal{P}_s: X \subset Z \subset Y, \\ & \text{if } Y \subset X \wedge \exists Z \in \mathcal{P}_s: Y \subset Z \subset X; \\ 3, & \text{if } X, Y \in \mathcal{P}_s \wedge X \cap Y = \emptyset \\ & \wedge (\exists Z_1 \in \mathcal{P}_s: X \subset Z_1 \wedge \neg \exists Q: (X \subset Q \subset Z_1)) \\ & \wedge (\exists Z_2 \in \mathcal{P}_s: Y \subset Z_2 \wedge \neg \exists T: (Y \subset T \subset Z_2)) \\ & \Rightarrow Z_1 = Z_2; \\ \geq 4, & \text{otherwise.} \end{cases}$$

A closeness metric for peer identifiers

This identifier space is not a metric space, as $\forall x, y, z \in \mathcal{I}_{\mathcal{P}}: d(x, z) \geq d(x, y) + d(y, z)$ and $\forall x, y \in \mathcal{I}_{\mathcal{P}}: d(x, y) = 0 \Rightarrow x = y$ are not true. The example of distance 3, where the following holds $d(x, y) = 3$ and $d(x, z_1) = 1$, $d(y, z_1) = 1$, and the fact that more services can be on the same geolocation, prove that fact. This closeness metric is used for management of the identifier space, graph embedding, and routing strategy.

Resources

The identifier space of a resource contains its geolocation and an optional string representing its metadata

The identifier space $\mathcal{I}_{\mathcal{R}}$ of resources $r \in \mathcal{R}$ in Globase.KOM contains geolocation loc (58 Bits) and an optional string meta representing the metadata (restaurants, emergency stations, etc. See Section 4.1.4).

The closeness metric $d_{\mathcal{R}}: \mathcal{I}_{\mathcal{P}} \times \mathcal{I}_{\mathcal{R}} \rightarrow \mathbb{N}$ for a peer X and a resource W with the identifiers x and w respectively is defined as

$$d_{\mathcal{R}}(x, w) = \begin{cases} 0, & \text{if } \text{loc}_x = \text{loc}_w; \\ 1, & \text{if } X \in \mathcal{P}_S, \text{loc}_w \in \text{zone}_x \\ & \wedge (\neg \exists Z \in \mathcal{P}_S): (\text{loc}_w \in \text{zone}_Z \wedge \text{zone}_Z \subset \text{zone}_x) \\ 2, & \text{if } X \in \mathcal{P}_S, \text{loc}_w \in \text{zone}_x \\ & \wedge \exists Z \in \mathcal{P}_S: \text{loc}_w \in \text{zone}_Z \subset \text{zone}_x \\ \geq 3, & \text{otherwise} \end{cases}$$

where z is the identifier of a peer Z .

The aim of this design decision is to enable deterministic routing and resolving queries based on peer and resource identifiers. The messages will be sent only to the peers that are either responsible for the queried resources or are in the shortest route path. Both efficiency and retrievability can be improved by this decision. The relation between the identifier space of peers and resources indicates structured overlays.

4.3.2 Mapping of Entities to the Identifier Space

Peers are responsible for resources in their location

Identifier part loc is given according to geolocation of peers and resources. In the case of mobile peers, identifier part loc represents the location of the peer when joining the network. As mobile peers do not provide services, their location is not crucial for indexing. Initially, zone is zero, as a newly joined peer is always treated as regular peer. Upon promotion of a regular peer to a superpeer, its identification part zone is defined according to the assigned zone of the responsibility. This identifier change implies an update of routing tables, which is discussed in Section 4.3.4. Random part rand of a peer identifier is defined in a joining process by its responsible superpeer. The process of bootstrapping and joining is described in Section 4.4. An optional metadata description m of the resource identifier is defined by a peer upon joining. Summarized, the mapping $\mathcal{F}_{\mathcal{P}}: \mathcal{P} \rightarrow \mathcal{I}_{\mathcal{P}}$ that associates peers with the identifiers from $\mathcal{I}_{\mathcal{P}}$ has the following characteristics:

- It is *complete* as all peers are associated with an identifier all the time,
- Mapping is *one-to-one* meaning $\forall x, y \in \mathcal{I}_{\mathcal{P}}: x \neq y \Rightarrow \mathcal{F}_{\mathcal{P}}(x) \neq \mathcal{F}_{\mathcal{P}}(y)$,
- Mapping is partially dynamic as the identification of a peer changes once it becomes a superpeer.

Resources are mapped to the resource identifier space $\mathcal{I}_{\mathcal{R}}$, using function $\mathcal{F}_{\mathcal{R}}: \mathcal{R} \rightarrow \mathcal{I}_{\mathcal{R}}$ where a peer with component *loc* in its identifier is responsible for the resource with the identifier containing *loc* as the location component. As resources need to be uniquely identified, the mapping $\mathcal{F}_{\mathcal{R}}$ is injective. This mapping implies the semantical closeness between resources and the corresponding peers.

4.3.3 Management of the Identifier Space

When bootstrapping the system, just one zone exists, covering the entire world, which is assigned to the first superpeer. Peers with high CPU power, good network connection, and a history of long online times, are marked as potential superpeers. As the network grows, highly loaded areas are clustered into rectangular zones using a clustering algorithm and assigned to one peer in that area, which becomes a superpeer by taking over the zone. The process of forming zones is described in Section 4.5. As described in Section 4.1.4, it is assumed that each peer provides information about the services related to their geolocation; however, superpeers are responsible for caching all resources stored in the *zone of the responsibility*. Thereby even if the peers are offline, basic information about their resources will be available at their superpeers.

Superpeers are responsible for peers in their zone

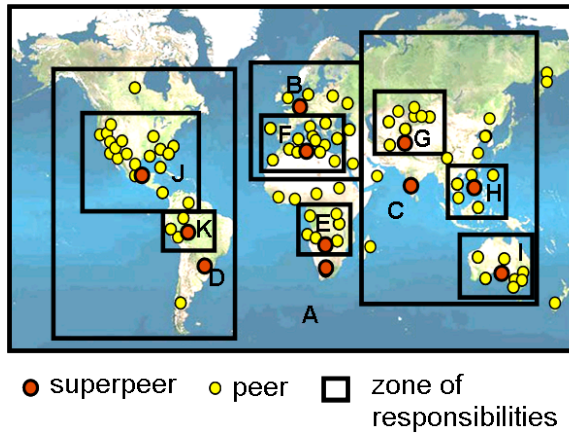


Figure 29: Management of the resource identifier space

Figure 29 depicts the resource management among the superpeers. For example, superpeer B manages resources in the zone containing Europe and superpeer E in the zone around the south of Africa. However, superpeer B does not manage the inner zone assigned to superpeer E.

Summarized, the function describing the management of the identifier space in Globase.KOM $\mathcal{M}: \mathcal{I}_{\mathcal{P}} \rightarrow \mathcal{P}^{\mathcal{E}}$, assigns each peer the responsibility for the set of identifiers $\mathcal{F}_{\mathcal{R}}(r) \in \mathcal{I}_{\mathcal{P}}$ of the resources r . It is *complete* as each identifier is being managed. It is effected by proximity, meaning that for a resource on a given location loc , responsible peers are:

- peer X_P on the same location, loc_{X_P} and
- the corresponding superpeer X_{P_S} , closest to the resource according to the closeness metric d_R

$$X_{P_S} \in \mathcal{M}(i) \Rightarrow d_P(\mathcal{F}_{\mathcal{P}}(X_S), i) = \min_{Y(\neq X_P) \in \mathcal{P}_S} (d_{\mathcal{P}}(Y), i).$$

Therefore, \mathcal{M} is *injective*, one-to-one, with the cardinality $|\mathcal{M}(i)| = 2$; two peers are responsible for one resource. It changes dynamically as the new zones are being formed and peers in them are assigned to a new responsible superpeer.

4.3.4 Graph Embedding

A basic neighborhood corresponds to tree node relations

A crucial requirement for the design of peer-to-peer location-based search is complete retrievability of the location-based searches, such as area search. Having peers as resources (Section 4.1.4) implies a close relation between the design of routing topology (graph embedding) and indexing structure (management of identifier space). The figure 29 is a valid depiction of both the management of indexing space and the basic routing topology of Globase.KOM, which can be modeled as a superpeer-based tree. Each zone is assigned to a superpeer located inside the zone, which keeps overlay and underlay contact addresses to all peers in that zone. Superpeers form a tree where peer A is called the parent of peer B, when B's zone is inside A's zone. Figure 30 shows the views of superpeers on different levels.

The basic routing topology is enhanced with the interconnections. Each superpeer maintains connections to other superpeers besides its parent and children. Also, each peer caches the contact information of other superpeers besides its parent. The main purpose of interconnections is to balance the load as they are shortcuts from one branch to the other branches of the tree. Thereby the routing through higher levels of the tree is avoided whenever it is possible. Additionally, bypassing the root superpeer makes query responses more efficient, especially in the case of a degenerated tree. There are two kinds of interconnections:

In order to provide long-range contacts, a peer maintains interconnection lists

- *Wide interconnections*, that connect two superpeers from different branches (distance 3 between their identifiers). Using wide interconnections, routing bypasses the peers in the higher level of the tree, improving load balancing and providing faster query resolution.
- *High interconnections*, that connect two superpeers in the same branch (distance 2 between their identifiers), help in failure recovery process.

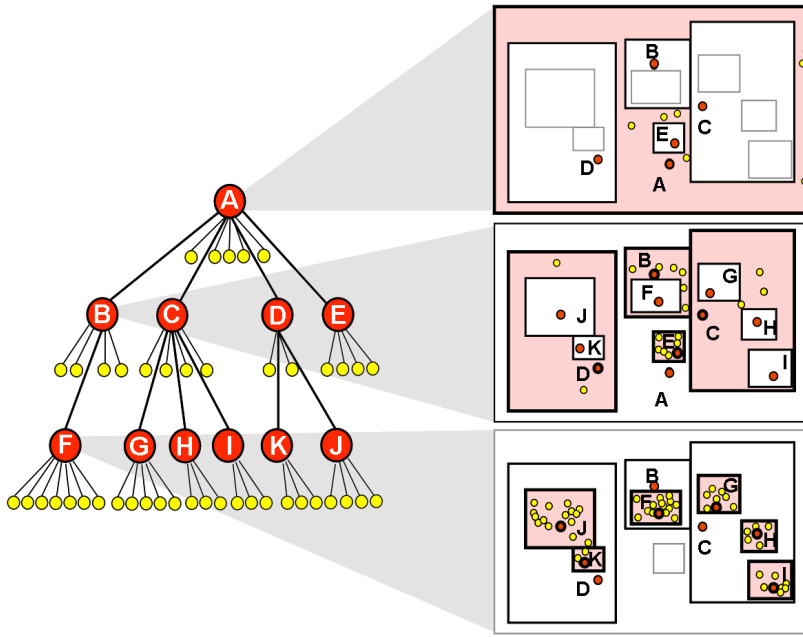


Figure 30: Basic routing topology of the Globase.KOM

Reiter [RSW05] presented an algorithm for constructing a fault-tolerant communication structure out of a core tree structure, where each peer initially only knows its parent and children. Their focus is the construction of an expander graph from a tree, using a random walk for collecting new edges such that the node in the graph have node degrees close to some constant. Tree reconstruction after failures is done by using new edges and heavily relies on the root of the tree.

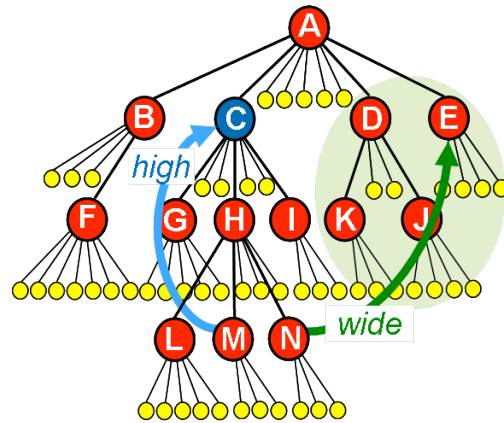


Figure 31: Interconnections

Our approach modifies Reiter's approach to avoid relying on the root superpeer in tree reconstruction, during failure recovery (explained in Section 4.3.6). Interconnections are used, which can direct to new

No active collection of interconnections, but peers learn from the received messages

parents in the tree. As a random walk introduces additional protocol overhead and traffic, the peers in Globase.KOM learn about new contacts through received messages (with similar maintenance costs as in [MMo2]). Additionally, interconnections are collected from the parent superpeers and other interconnections.

Each query message includes the address of the query initiator and the address of the responsible superpeer (for detailed description of messages in Globase, see Appendix H). Upon receiving a message, each superpeer or peer checks if the initiator is its parent or child and if it is part of its subtree. Checking is done with a simple calculation of the described zone in the sender's ID. If the sender is not a parent or a child, then the recipient adds the sender to its interconnection list. The size of an interconnection list as well as their effect on the performance and costs are investigated in Section 6.1.2.

Each zone has a replica superpeer

In order to have an efficient failure recovery, each zone has a *replica superpeer*. That is a peer, marked as potential superpeer which is the closest to the geometrical center of the zone. It receives the complete routing table of the superpeer responsible for the zone with periodic updates (see Section 4.3.6).

In summary, the routing topology can be modeled as a symmetric directed graph $G = (\mathcal{P}, \mathcal{E})$, where \mathcal{P} is set of vertices (or peers), and \mathcal{E} set of edges, connections. Although the core structure of Globase.KOM is a tree, as interconnections introduce cycles, the resulting structure is not a tree. Each peer maintains a routing table, a set of connections to the *neighboring* peers $\mathcal{N}(p)$ where $\mathcal{N}: \mathcal{P} \rightarrow 2^{\mathcal{P}}$.

Formal description of the graph embedding

Each superpeer $X \in \mathcal{P}_S$ maintains the contact addresses of:

- the peers inside its zone of responsibility, excluding the inner zones, $\forall Q \notin \mathcal{P}_S: d_P(F_P(X), F_P(Q)) \geq 1$
- superpeers responsible for inner zones (children in the tree),
- the parent node in the tree $\forall Q \in \mathcal{P}_S: d_P(F_P(X), F_P(Q)) = 1$,
- the root superpeer $R \in \mathcal{P}_S$,
- interconnected superpeers $I \in \mathcal{P}_S: d_P(F_P(X), F_P(I)) \geq 3$,
- brother superpeers (with the same parent superpeer) in the case of direct children of the root superpeer,
- the replicated superpeers, and
- a cache list of recently contacted peers.

Each regular peer maintains the following contact addresses:

- the parent superpeer $Q \in \mathcal{P}_S: d_P(F_P(X), F_P(Q)) = 1$,
- the root superpeer $R \in \mathcal{P}_S$,
- interconnected superpeers $I \in \mathcal{P}_S: d_P(F_P(X), F_P(I)) \geq 3$,
- a cache list of recently contacted peers.

The routing table is not always symmetrical, meaning that if a peer X maintains the contact address of a peer Y (Y is a neighbor of X) it does not necessarily imply that the routing table of peer Y contains the contact information of peer X (X is a neighbor of Y). Connections between superpeers are symmetrical. Connections between peers and connections to superpeers, that are not direct parents, are asymmetrical.

As we can see, the neighboring relationship is tightly connected with the distance function defined in 4.3.1 in order to enable efficient routing. Peers are connected to their immediate neighbors (parent and children peers)

$$\forall X, \forall Y \in \mathcal{P}: d_P(F_P(X), F_P(Y)) < 3 \Rightarrow Y \in \mathcal{N}(X)$$

which brings the *local connectivity* of the graph. To improve robustness, each peer maintains a subset of *long-range contacts* $X, Y \in \mathcal{P}, d_P(F_P(X), F_P(Y)) > 3$, such as wide interconnections and connection to the root superpeer.

Local and long-range contacts are selected according to the closeness metric

4.3.5 Routing Strategy

Corresponding to the functional requirements described in the Section 1.2.2, the primary goal of Globase.KOM is finding the peers that are responsible for the resources r which are described by metadata $meta$ and:

- located in the location L (*lookup*),
- closest to the location L (*find the closest*), or
- inside of the area A (*area search*).

Routing strategy for area search, find the closest, and lookup

Location L is described by a geolocation representation (see Section 4.1.1), and area A is described by the geolocation of its center C and radius r .

The routing strategy for these three types of queries is different and will be discussed separately.

Lookup

The result of the *lookup* for the resource r described with the location L_r and metadata $meta_r$ will be found on peers $\mathcal{F}_{\mathcal{R}}(r)$:

- set of peers $\forall X \in \mathcal{P}, x = \mathcal{F}_{\mathcal{P}}(X):$
 $(loc_x = L \wedge meta_{\mathcal{F}_{\mathcal{R}}^{-1}(x)} = meta)$
- \emptyset if $\neg \exists X \in \mathcal{P}, x = \mathcal{F}_{\mathcal{P}}(X): (loc_x = L \wedge meta_{\mathcal{F}_{\mathcal{R}}^{-1}(x)} = meta)$

The routing strategy in Globase.KOM is greedy, meaning that the query is forwarded to the neighbor which is closest to the destination, peer $\mathcal{F}_{\mathcal{R}}(r)$ (according to the closeness metric d_R described in Section 4.3.1). After checking for matching peer in all connections of the initiator of the query, the lookup message, which includes the address of the

initiator, is forwarded to an interconnection I where $d_R(\mathcal{F}_P(I), r) \leq 2$ or if there are no such, further upwards in the tree, using parent-connections. This repeats on each superpeer in the route, which receives the lookup message, until $\mathcal{F}_R(r)$ is reached.

Area Search

The result of the *area search* for the resources r in the area A , described by metadata meta_r , will be found on peers $\mathcal{F}_R(r)$:

- set of peers $\forall X \in \mathcal{P}, x = \mathcal{F}_P(X)$:
 $(\text{loc}_x \in A \wedge \text{meta}_{\mathcal{F}_R^{-1}(x)} = \text{meta})$
- \emptyset if $\neg \exists X \in \mathcal{P}, x = \mathcal{F}_P(X): (\text{loc}_x \in A \wedge \text{meta}_{\mathcal{F}_R^{-1}(x)} = \text{meta})$

Routing for area search is recursive, more superpeers send matching results

The final routing destinations of an area search message are the following superpeers: $\forall X \in \mathcal{P}_S: \text{zone}_x \cap A \neq \emptyset$. The routing here is similar to the routing strategy in lookup. Each superpeer in the routing path however, checks if his zone intersects the area A given in the area search query. If it intersects, the superpeer sends to the initiator of the query the list of the peers from the intersecting area. The search is considered finished after a specific timeout. Simulation studies 6.1 showed that the optimal value for this timeout is 2 seconds. For each received message, interconnections are updated as described in Section 4.3.4. An example of an area search is given in Figure 32. A peer in

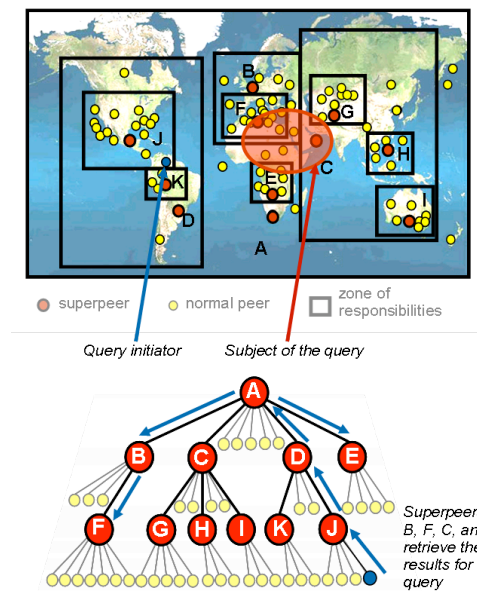


Figure 32: Example of an area search

the zone of superpeer B sends an area search message containing a description of the marked zone. As the zone does not intersect the zone superpeer B is responsible for, the area search message will be forwarded to the superpeer A. This process continues on each superpeer

in the path. Eventually, superpeers A, C, and D will reply with the list of the matching results.

Find the closest

The result of the *find the closest* query for the resources r described by the location L and metadata meta_r , will be found on exactly one peer $\mathcal{F}_{\mathcal{R}}(r)$:

- $\exists X \in \mathcal{P}, x = \mathcal{F}_{\mathcal{P}}(X):$
 $(d(\text{loc}_x, L) = \min_{Y \in \mathcal{P}} (d(\text{loc}_y, L) \wedge \text{meta}_{\mathcal{F}_{\mathcal{R}}^{-1}(x)} = \text{meta}))$
- \emptyset if $\neg \exists X \in \mathcal{P}, x = \mathcal{F}_{\mathcal{P}}(X): \text{meta}_{\mathcal{F}_{\mathcal{R}}^{-1}(x)} = \text{meta}$

where d is the geographical distance between two points.

An example of the routing strategy for a *find closest peer* query is depicted in Figure 33. The initiator of a query checks for matching peers from its contacts and calculates the distance $d_0 = (L, p_0)$ to the closest one p_0 , if any are found. The find closest message (which includes d_0) is then forwarded to the superpeer $S_1 \in \mathcal{P}_S: L \in \text{zone}_{S_1}$, whose zone of responsibility contains the location L . If it finds a closer matching peer $p_1 \in \mathcal{P}: d(\text{loc}_{p_1}, L) = d_1 < d_0$, then it calculates the distance d_{border_1} from the location L to the closest border of its zone. If $d_1 > d_{\text{border}_1}$ then the superpeer forwards the message (including d_1) to its parent superpeer S_2 . It looks for a matching peer p_2 in its zone and calculates its distance $d_2 = d(L, p_2)$ to the given location L . Further, the distance d_{border_2} between the border of zone S_1 and its closest brother S_3 is calculated. If $d_2 > d_{\text{border}_2}$ then the superpeer S_3 is queried for matching peers. If the distance d_3 from the matching peer p_3 found in the zone of S_3 is smaller than d_2 , then the resulting peer p_3 is retrieved to the initiator of the query.

Routing for find the closest is also recursive

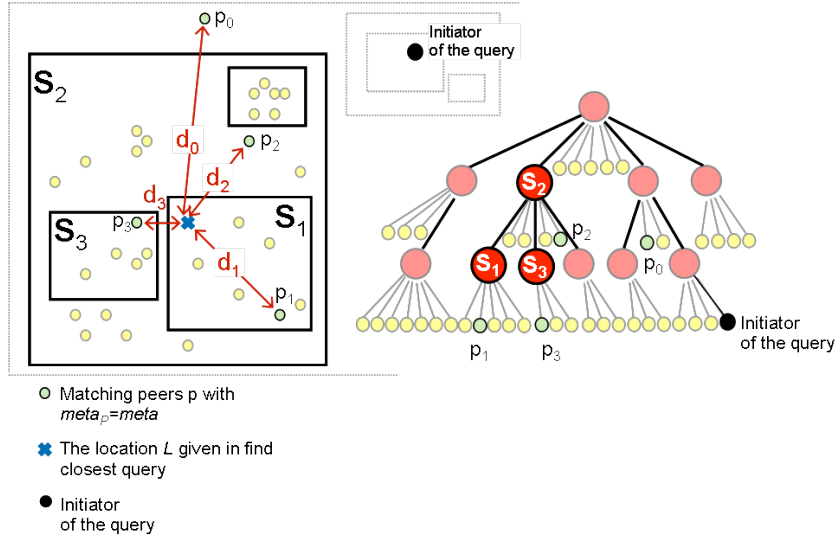


Figure 33: Example of the *find the closest* query

Otherwise,

- if $d_1 > d_0$ then p_0 is the final result,
- if $d_1 < d_{\text{border}_1}$, then p_1 is the result,
- if $d_2 < d_{\text{border}_2}$, then p_2 is the result.

The number of hops in the resulting route of a *find the closest peer* query depends on the distance $d(\text{loc}_{\text{initiator}}, L)$ and on how many innerzones are involved in the query. For example, if there was a child-zone of the zone of superpeer S_1 or S_3 which has matching peer closer to given location L , the number of involved superpeers would increase.

4.3.6 Maintenance Strategy

Maintenance strategy is proactive

Autonomy of peers in peer-to-peer overlay networks implies high dynamics of peer participation. As each peer in Globase.KOM is both server and a client at the same time (with the exception of mobile peers), their participation directly influences on the availability of the resources and connectivity of the network. Maintenance of contacts and resources are crucial for all quality properties, besides robustness and stability (see Section 1.2.3). As discussed in the previous chapter (particularly in Section 3.3.6), the maintenance strategy consist of *failure detection* and *failure recovery*. After the failure of a peer is detected (using failure detection mechanisms), suitable mechanisms are used to keep the network connected and resources available (using failure recovery mechanisms).

Failure detection mechanism is not the same for all connection types

Globase.KOM has seven main types of the neighbor-relations:

1. parent \leftrightarrow child (superpeer \leftrightarrow superpeer),
2. superpeer \leftrightarrow and its interconnections (superpeer \leftrightarrow superpeer),
3. superpeer \leftrightarrow replicated superpeer (superpeer \leftrightarrow superpeer),
4. child peer \leftrightarrow parent (a regular peer \leftrightarrow its parent superpeer),
5. a regular peer \rightarrow interconnected superpeer (a regular peer \rightarrow superpeer),
6. peer \rightarrow root superpeer (peer \rightarrow root superpeer), and
7. peer \rightarrow cached connection (peer \rightarrow peer).

Each of these neighbor connections has different importance for the robustness of the overlay. For example, failure of a regular peer does not affect routing as the routing strategies use mainly superpeer connections. Cache connections are also not crucial for the robustness: However, the connection between a superpeer and its replica is valuable in keeping the peers connected. Additionally, each connection type is exposed to different dynamics of failures. Connections between superpeers fail less frequently than the connections that involve regular peers. Therefore, our design of failure detection and recovery mechanism considers all types of the connections separately.

Failure Detection

Globase.KOM employs both active and passive, baseline and sharing failure detection. We use both, a gossip and probe approach, i.e. sending periodical message 'I am alive' or 'Are you alive?' respectively.

In the case of the *connections between two superpeers* (cases 1-3), periodical probing with sharing is used. After missing a sequence of three probe messages a failure is detected. The negative liveness information is then shared downwards in the tree (using parent \leftrightarrow child superpeer connections). In that way regular peers are also being informed about failures of their high interconnected superpeers. A probing frequency must be high in the case of direct parent/child superpeer and replicated superpeer connections. These connections build the basic tree overlay structure which is maintained with the highest priority.

A connection between superpeers is periodically probed

A *superpeer* sends periodically 'I am alive' liveness information to its *regular peers* it is responsible for (case 4). After missing a sequence of three such messages, a peer detects the failure of its parent superpeer. The superpeer receives periodically liveness information from a regular peers, too. However, the frequency of these messages are significantly smaller than that of the messages from the superpeer. The reason for this choice is that liveness information of peers is only necessary for some cases of *find closest queries*. A superpeer maintains the basic information of its peers and sends a matching subset of them to an initiator of a query. Eventually, the initiator establishes a contact with the particular peer and learns about its liveness. A failure detection of a child peer \leftrightarrow parent connection is therefore an active, baseline using gossip approach.

A superpeer sends its liveness information periodically to its child-peers

Failures of connections between *a regular peer and an interconnected superpeer or cached contact* (case 5 and 7) are passively detected. Once a peer needs to use its cached contact, it piggybacks a probe with an explicit acknowledgment. A regular peer does not need to actively probe its interconnected superpeer as it finds that liveness information out from its parent superpeer.

Passive failure detection for a regular peer \leftrightarrow its interconnection or cached contact

Connection peer \rightarrow root superpeer (in the case a peer is not the direct child of the root superpeer) is probed passively, as this negative liveness information (as well as information about the new root superpeer) is shared to all peers downwards in the tree. The reason for this choice is reducing redundant burst of traffic to the root superpeer.

A rootsuperpeer \leftrightarrow a regular peer probed passively, negative liveness information shared

Failure Recovery

The goal of failure recovery is to replace connections to failed peers with the appropriate connections that preserve the regular routing method. Failure recovery begins once a failure is detected. The applied failure recovery mechanism depends on the connection type.

When a superpeer detects a *failure of its regular peer*, it will just mark it as offline, but will not remove it from its contact list. It is very likely that a failed peer will appear online on the same location, therefore it will keep the same identifier and belong to the zone of the same superpeer when it rejoins. Mobile peers, as already discussed in Section

4.1.3, are just clients in the system and therefore their maintenance is irrelevant.

A replica peer takes over the zone upon the failure of its superpeer

Each superpeer chooses a replica peer from the list of superpeer candidates. To avoid long-distance traffic (introducing the additional traffic between ISPs), the replica peer is the geographically closest peer, which fulfills requirements to become a superpeer. Initially, a superpeer sends to its replicating peer all its contacts (e.g. regular peers, parent/children superpeer, interconnections) and then periodically (within a probe message) sends updates. Once *a failure of a replica superpeer* is detected, a superpeer simply chooses a new replica peer. If *a superpeer fails*, its replicating peer takes over its zone (all of its children peers too) and informs all of its contacts about it.

High interconnections used to reach new superpeer when the replica peer fails together with its superpeer

In the case that both *superpeer and replica peer fail*, a child peer uses the high interconnections (or root superpeer connection, in the worst case) to route the appropriate message to the smallest zone containing the location/zone of the sender. The reached superpeer (most likely the parent of the failed superpeer, if it is still online) adds the peer to its peer list or sets the superpeer as a new child superpeer and informs it that it is a new parent superpeer. This mechanism also works well in the case when multiple superpeers fail simultaneously, as the peers contact either interconnected superpeers or the root superpeer for recovery. If none of its contacts are alive, the superpeer/peer will simply rejoin the network.

Failure recovery when the root superpeer fails

A special case of superpeer failure is certainly the *failure of the root superpeer*. In the case that its replica superpeer fails, one of the child superpeers will take over the responsibility of the failed root superpeer. In order to avoid forming several independent trees, the *Election on Bully* algorithm [GM82] is applied. Therefore, all child superpeers of the root superpeer keep *brother* connections to each other. Using the election algorithm, when a child superpeer notices the failure of the root superpeer, it starts the election where it chooses the brother with the highest ID and sends him an election message with a sequence number. As soon as the election is finished, the elected superpeer takes over the zone of the root superpeer and sends information about the new root superpeer to all superpeers in the tree.

If an *interconnected or cached superpeer fails*, a peer will simply remove it from its contact list and replace it with an appropriate connection from future message exchange.

Leaving the network (going offline and sending a 'good-bye' message) is also assumed by the design of Globase.KOM. When leaving the network, a superpeer informs its replica superpeer which takes over the responsibility for all contacts from the leaving superpeer and informs them.

4.4 BOOTSTRAPPING

Joining the peer-to-peer network is an essential operation which must be always successfully completed. If a peer rejoins the network, it can use the connections it previously had in its routing table. However,

if the new peer wants to join the network, whom should it contact? This problem is called *bootstrapping* problem. An overview of bootstrapping techniques currently used in peer-to-peer applications is given in [KWSW07].

Globase.KOM's bootstrapping procedure uses a mediator approach. A list of so-called *bootstrapping peers* is stored on a *mediator* – a server (similar to the host cache in Gnutella 0.4) whose URL is hard-coded into the bootstrapping protocol. The bootstrapping peers are regular participants of the overlay that are expected to have static IP addresses (or domain names) and high availability. Upon joining, a peer contacts the mediator and receives the list of available bootstrapping peers. It contacts one of the available peers and sends its geolocation. The contacted bootstrapping peer routes an appropriate message to the superpeer who is responsible for the zone the joining peer is located in. Routing is done in the same manner as lookup 4.3.5. The responsible peer forms the identifier (like described in Section 4.3.1) for the new peer and adds it to its peer list. It sends back a message with the formed identifier, the contact information of the root superpeer and its interconnection list. Afterwards the joining process is finished. In the case of the peer being the first to join, the process is described in Section 4.5.

*A mediator approach
for bootstrapping*

4.5 FORMING THE ZONES

The first zone in the overlay covers entire world. It is assigned to the first peer joining the overlay, which must be capable of becoming a superpeer (high capacity, spare bandwidth, good network connectivity, publicly reachable). As new peers join, the load of the first superpeer grows until it reaches a load threshold (explained shortly). Then a highly loaded area, with the dense distributed peers is clustered into a new rectangular zone. It is then assigned to one peer in that area (marked previously as a potential superpeer), which becomes a superpeer by taking over the zone. In the following we will explain load thresholds and the process of forming new zones.

*New zones are formed
when a load of a peer
reaches a threshold*

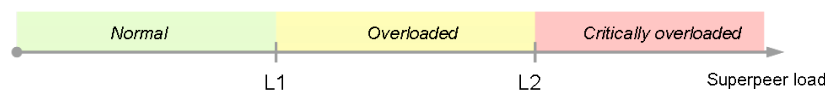


Figure 34: Load levels and thresholds in Globase.KOM

As a metric for the load of an area, we use the number of peers connected to a superpeer, as this directly influences the number of query messages a superpeer receives and how many contacts it must maintain. There are three load levels - *normal* (below a threshold L_1), *overloaded* (between thresholds L_1 and L_2), and *critically overloaded* (above L_2). Once a superpeer's load exceeds the threshold L_2 , it runs a *single linkage clustering* algorithm (explained in Section 4.5.1) in order to create a new zone inside its own zone. The new zone is then assigned

*Two load threshold
levels, load = number
of peers in the zone*

to one of the peers within the formed zone which had been marked as a potential superpeer.

A new zones must fulfill the following requirements:

How should a zone look?

- to be rectangular,
- to contain approximately $L_2 - L_1$ peers, and
- not to overlay with the existing zones.

The following information is needed to create a new zone:

Information needed to create a new zone

- the position of all peers inside the overloaded superpeers' zone of responsibility,
- the position of the overloaded superpeer,
- a list of zones of all child superpeers of the overloaded superpeer,
- the target size for the new zone (the difference of L_1 and L_2), and
- the hotspot criterion R_1 (the share of dense distributed peers, needed to qualify their area as a hotspot).

Forming a new rectangular zone which contains hotspot with approximately $L_2 - L_1$ peers and do not overlap with the existing zones, contains the following three steps:

Three steps in forming the zone

1. finding a hotspot (using single linkage clustering algorithm),
2. resolving zone overlapping, and
3. extending zones to target size.

These steps will be explained in detail in the following sections.

4.5.1 Finding a Hotspot

A hotspot describes an area on the world map, where a high number of peers are situated. It is, therefore important to consider hotspots, when creating new inner zones to reduce the load for an overloaded superpeer. The number of peers that form a hotspot is specified by the hotspot criterion R_1 . To find hotspots within an existing zone, Globase.KOM uses *single linkage clustering*, an agglomerative hierarchical clustering method [Scho8]. It works in the following way.

Single linkage clustering algorithm used for identifying a hotspot

Every peer that is located within the zone of the overloaded superpeer is treated as its own cluster in the initial step of the algorithm. A proximity matrix PM of size $(n \times n)$ is calculated (where n is the number of clusters). The entry $PM_{i,j}$ of the proximity matrix PM denotes the distance between cluster I and cluster J . For single linkage, the distance $dist_{I,J}$ between the two clusters I and J is calculated as follows:

$$dist_{I,J} = \min_{\substack{x_i \in I \\ x_j \in J}} (dist(x_i, x_j))$$

The distance function $\text{dist}(x, y)$, used in the formula above, calculates the Euclidean distance between two coordinates x and y .

In the next step, the two clusters with minimal distance between each other (in this example referred to as clusters K and L) are merged together into a new cluster PM_1 . Rows and columns for both K and L are deleted from the proximity matrix, before a new row and a new column for the newly formed cluster M is added to PM . The distances between cluster PM_1 and all other clusters in the proximity matrix have to be calculated and inserted in PM . This step is repeated until the number of peers in the largest cluster meets the hotspot criterion R_1 .

The reasons for choosing single linkage clustering algorithm are the following:

- Hierarchical (especially single linkage) are more versatile than the partitional algorithms [JMF99].
- It does not require the number of clusters in the zone to be known in advance [KP04].
- It makes less compact clusters than the complete linkage algorithm [JMF99]. In our case it is important to avoid the zones being too small as it can result in a very deep tree. Room for future hotspots needs to be provided in resulting inner zones.
- Despite of its time complexity $O(N^2 \log(N))$, it is suitable as N (number of the peers in the zone) is not too big ($L_2 + 1$) and identification of a hotspots does not occur often.

The reasons for choosing single linkage clustering algorithm

4.5.2 Resolving Zone Overlapping

After a hotspot has been identified in the first step of the algorithm, the smallest possible zone, containing all peers of the hotspot, is formed. Since the algorithm that identifies hotspots only considers the distances between peers, the provisional zone needs to be checked against intersections with other zones existing inside the zone of the overloaded superpeer. Figure 35 shows an example where the minimal zone created for the hotspot found in the first step of the algorithm intersects with the existing zone of a superpeer.

In this case the overlap of the new zone around the hotspot (shown as the grey rectangle in Figure 35) and the existing zone is resolved by removing peers from the hotspot area. To minimize the number of peers that need to be excluded from the hotspot, the number of peers in the four areas around the existing zones are evaluated. The area containing the most peers of the hotspot defines the new non-intersecting zone for the hotspot. In Figure 35 the area to the left of the zone of superpeer SP_1 contains most peers of the original hotspot (28 peers in comparison to 20 in the area above the existing zone). The resulting non-intersecting zone for the hotspot is shown in Figure 36.

New zones must not overlap with the existing zones

This step, resolving intersections between two zone, needs to be repeated for all existing zones within the overloaded superpeers zone.

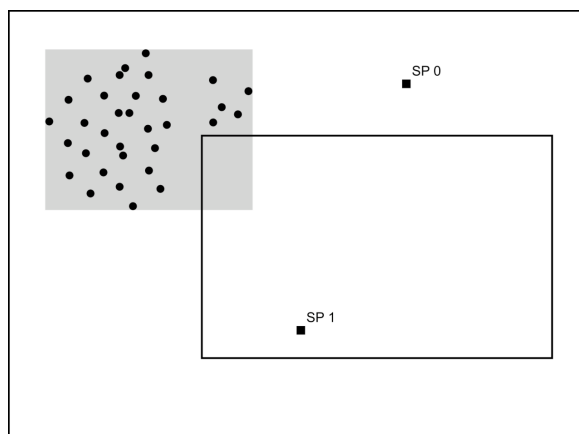


Figure 35: Overlapping zones during clustering

4.5.3 Extending a Zone

After a hotspot has been found and possible overlapping with existing zones have been removed, more peers are added to the new zone until it contains the targeted number of peers. This target is determined by the difference of the two load thresholds L_1 and L_2 . Using the distance of a peer to the center of the zone as metric, the closest peer is added to the hotspot zone, if no intersections with existing zones occur.

A zone must include a certain load

To ensure that well-formed zones are created by the clustering algorithm, another criterion needs to be met. Only if the ratio between the length of long and short edge of the zone remains below 2, the afore chosen closest peer is added to the zone. Should the zone already be deformed (corresponding to a ratio larger than 2), peers are only added, if the resulting zone has a smaller ratio of long to short edge than the resulting zone from the previous step. This step is repeated until the number of peers inside the new zone reaches the targeted size ($L_2 - L_1$) or no more peers can be added to the zone.

Using the presented algorithm for forming the zones, we create a zone which fulfills the stated requirements: rectangular zone, containing approximately $L_2 - L_1$ peers, which does not overlap with the existing zones.

4.6 SUMMARY

Globase.KOM is a hybrid, structured, tree-based overlay

In this chapter, we presented our peer-to-peer overlay network for location based search, Globase.KOM. Design decisions and crucial components of the overlay are presented and discussed according to the reference model given in [LAG⁺05]. Lessons learned from the design of current peer-to-peer overlays are reflected in the decisions made, based on the requirements stated in Section 1.2.

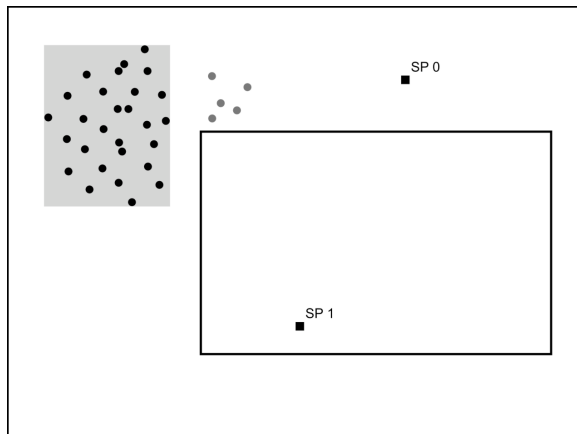


Figure 36: Overlapping of zones resolved

The resulting peer-to-peer overlay is a hybrid, structured, tree-based overlay network enhanced with interconnections. By introducing different roles of superpeers and regular peers, according to their capacities, connectivity and online behavior, we support heterogeneity among the participants. Regular peers are directly connected to the superpeers (noting that all peers offer and consume the services, having both server and client role in the overlay), while superpeers are connected in a structured overlay in a form of tree with varying number of children. The world projection is divided in rectangular, not overlapping zones. Each zone is assigned to a superpeer located inside that zone and keeps overlay/underlay contact addresses to all peers in that zone. Superpeers form the tree where peer A is called the parent of peer B when B's zone is inside A's zone. A peers identifier contains its geolocation, zone of responsibility and random part supporting more than one peer on the same location. This allows for more efficient routing and calculations as it is not necessary to contact a peer to find out its location or zone of the responsibility. Additionally, it enables deterministic routing and resolving queries based on peer and resource identifiers. Messages are sent only to the peers that are either responsible for the queried resources, or are in the shortest route path. Both efficiency and retrievability are the goal of this choice. The relation between identifier space of peers and resources implies a structured type of overlay. Robustness and stability are addressed by multiple failure detection and failure recovery mechanisms, depending on the type of the connection and its importance in overlay. In that way, additional overhead is avoided.

The evaluation of the presented overlay, regarding the stated requirements is the focus of the following sections.

Rectangular, not overlapping zones assigned to the superpeers

A peer identifier contains geolocation

Part III

EVALUATION

A crucial step in the design of a peer-to-peer overlay network is proving whether the presented solution fulfills the given requirements. Additionally, the properties of the designed solution can only be understood through exhaustive evaluation under varying conditions. It is also the crucial step in the design process, as it provides necessary feedback. In order to carry out the evaluation correctly, the following questions should be addressed:

- What are the goals of the evaluation? What exactly do I want to address?
- What are the metrics and criteria of the evaluation? How can we quantify the quality of the solution?
- How do we measure whether the solution fulfills the given requirements?
- What workloads should be used in the experiments?
- Are there any variable parameters in the solution? What are their effects on the quality of the solution?
- What evaluation technique and tool is the most appropriate for grasping all aspects of the overlay behavior?

In the following sections we answer these questions. The evaluation goals are stated in Section 5.1. Basic and derived metrics are presented in 5.2 and measurements of quality properties, described in non-functional requirements (Section 1.2.3) are defined in Section 5.4. Workloads used are described in Section 5.3. A discussion about the appropriate evaluation technique is the subject of Section 5.5.

5.1 GOALS OF THE EVALUATION

The most obvious goal of the evaluation is to prove that the solution fulfills given functional (1.2.2) and non functional requirements in Section 1.2.3. Additionally, it is crucial to show the influence on the quality of the variable parameters of the protocol and identify the most optimal values for the various scenarios. The goal of the evaluation of Globase.KOM is thus answering three questions:

1. Does the solution address all functional requirements?
 - Finding all peers or resources in a given geographical area,
 - Finding all peers or resources in a specific geolocation,
 - Finding the peer or resource closest to a given geolocation.

What are the questions to be answered before the evaluation of a solution?

The evaluation should provide answers to three crucial questions

Fulfilling functional requirements?

*Efficient, scalable,
stable, robust, and
valid results?*

2. Does the solution address all non-functional requirements?
 - Efficiency, as the ratio of performance and costs.
 - Stability, as the ability to maintain the quality of offered services and functions under changed system environment and conditions (e.g. intensive regular overlay operations).
 - Robustness, as the ability to recover and maintain the quality and functions when critical failures occur.
 - Scalability, as the quantitative adaptability of the system to a changing number of participants or services in the overlay, while preserving the performance.
 - Validity, as the ability to retrieve all matching results, which are up to date.

*Influence of the
parameters on the
overlay behavior*

3. How do the parameters affect the performance?
 - The **load threshold** parameters L_1 , L_2 might influence the size and shape of the superpeer-tree and the distribution of load within the network.
 - The **number of interconnections** should be large enough to, for example, allow a peer to remain connected when its direct parent fails. At the same time a large number of interconnections can bring additional maintenance overhead. Setting this number should, therefore, consider the tradeoff between robustness and costs.
 - The **size of the cache** influences the response time, and thereby the performance of the overlay. It causes, however, tradeoffs between performance and costs, but it is also connected with the user behavior in the terms of requested resources. If it is likely that the resources will be requested again, caching makes sense. Otherwise, it just brings additional costs.
 - **Timeouts** of the operations have large influence on the retrievability and operation time, causing the tradeoff between validity and performance.
 - **Frequency of periodical failure detection messages** is crucial for robustness, but it must be set carefully, taking into account additional maintenance overhead it can cause.
 - Number of missing periodical liveness information before the failure is detected is obviously crucial for robustness. The faster the failure is detected, the faster it will be repaired. However, the possibility of false positive needs to be minimized.

The methodology of assessing the quality properties (non-functional requirements) needs to be defined. We cannot say that the solution is '60% robust' or '8 units scalable', but appropriate metrics, scenarios, and workloads must be set to address these quality aspects. Appropriate experiment design with a combination of metrics and workload are the only ways of addressing them and will be described in the Section 5.4.

5.2 METRICS

A metric is a measure for quantitatively assessing a quality property. Metrics that are chosen for the evaluation of Globase.KOM can be classified into *basic* and *derived* metric (which are a combination of two or more basic metrics). Furthermore, we can distinguish *micro* and *macro* metrics. Macro metrics measure the results for testing the functionality of the system, such as area search or finding the closest peer. Examples of macro metrics are response time or completeness of the results. Micro metrics, on the other hand, explain the system behavior (described by macro metrics) taking into account the design of the system. For example, a micro metric number of contacted peers in the routing path can explain the macro metric response time. Micro metrics are dependent on the examined overlay and may differ from one overlay to another.

As criteria for these two classifications are not related (the first is mathematical, the second is according to the layer of observation, measurement perspective), the classifications are independent and non-exclusive. In the following sections we present the basic and derived metrics.

Metrics can be classified into basic and derived, macro and micro

5.2.1 Basic Metrics

The basic metrics chosen for the evaluation of Globase.KOM are response time, number of contacted peers in the overlay and underlay route, number of the received and sent messages, consumed upload and download bandwidth per peer, depth and breadth of a tree, and number of the stale contacts and messages.

Basic metrics: response time, number of hops, received/sent messages, bandwidth consumption, etc.

- *Retrievability* (macro metric) describes the ability of the overlay (mainly depends on routing strategy) to retrieve all existing matching results. It is defined as the ratio:

$$\text{Retrievability} = \frac{N_{\text{retrieved}}}{N_{\text{existing}}}$$

where $N_{\text{retrieved}}$ is the number of the matching resources included in the query response and N_{existing} is number of existing matching resources in the overlay.

- *Response time* (macro metric) for query operations is the time span between the initiation of the query and the time point $t_{\text{res}} = \min(t_{\text{all}} - t_{\text{start}}, T_{\text{timeout}})$ where t_{all} is the point when all matching results are retrieved, t_{start} point of query initiation, and T_{timeout} is a timeout of the corresponding query (lookup, area search, find the closest). Here it is necessary to have a ‘global knowledge’ of the overlay to know when all existing matching results in the system are in fact included in the retrieved results. Only then can t_{all} be determined. Otherwise, for the case of area search, just the retrievability can be measured, and response time would be always equal to the operation timeout T_{timeout} . For

measuring the time of lookup and find the closest, where a user queries one particular resource, $t_{all} - t_{start}$ a ‘global knowledge’ is not needed. $T_{timeout}$ must be long enough to cover the worst case response times.

- *Number of overlay hops or hop count* (micro metric) for a query is the number of contacted superpeers in the routing for resolving the query. The measurement of this metric depends on the number of superpeers which retrieve the matching results. In the case of lookup or find the closest peer, just one superpeer replies to the query but more superpeers are involved. In area search, each superpeer whose zone of responsibility intersects the queried area, replies to the query.
- *Number of the received and sent messages* (micro metric) is one of the metrics that describes the load of each peer. Each sent or received message is counted and differentiated according to the message type (maintenances or query messages).
- *Consumed upload and download bandwidth* (micro metric) is a more precise metric for describing the consumed resources of a peer. We distinguish upload and download bandwidth and bandwidth used by different types of the messages.
- *Depth and breadth of a tree* (micro metric) is a special metric describing the Globase.KOM tree structure. The broader tree can lead to a decreased number of hops or query response times, but also to much bigger loads for the root superpeer. A very deep tree can increase the number of hops needed to resolve a query, response time, and message overhead. In order to grasp the effects of the tree shape on the quality, this metric is crucial. Breadth of the tree is the number of children of the root superpeer while depth is the number of superpeers in the longest branch.
- *Number of stale contacts* (micro metric) describes the freshness of the routing tables. It is the number the peers’ contacts that are not alive (i.e. not online). Slow failure detection, or recovery, cause a higher number of stale contacts and can lead to big stability and robustness issues.
- *Number of stale messages* (micro metric) further addresses the maintenance strategy of an overlay. It is a more critical metric for the robustness and stability than the number of stale contacts. It presents the number of messages sent to the offline peers.

5.2.2 Derived Metrics

*Derived metrics:
relative delay penalty,
load balance ratio,
recovery time, etc.*

Derived metrics are based on the previously present basic metrics. They are either mathematically connected (relative delay penalty, load balance ratio, stale message and stale contact ratio) or their measurement relies on the other, basic or derived metrics (recovery time).

- *Relative Delay Penalty (RDP)* (micro metric) describes how well the overlay structure matches the underlying network topology. It is defined as the ratio

$$RDP = \frac{t_{\text{overlay}}(A, B)}{t_{\text{underlay}}(A, B)}$$

of the measured latency introduced by sending a message from point A to B through the overlay structure $t_{\text{overlay}}(A, B)$ and the corresponding latency when sending it directly through the underlay $t_{\text{underlay}}(A, B)$ [JMW03].

- *Load Balance Ratio (LBR)* (micro metric) describes the fairness of the load balance in the overlay. the load (e.g. consumed bandwidth) should be evenly distributed, taking into account the heterogeneity of the peers in the overlay. The load balance ratio is defined as the ratio:

$$LBR = \frac{L_{\text{mostloaded}}}{L_{\text{median}}}$$

where $L_{\text{mostloaded}}$ presents the load of the most loaded peer and L_{median} is the load of median loaded peer. Metric for the load can be either the number of the received or sent messages or amount of bandwidth consumed.

- *Stale contacts ratio* (micro metric) is the metric which further describes the basic metric ‘number of stale contacts’ (N_{sc}). Ratio R_{sc} relates this metric to the overall size of the routing table (N_{allc}), in order to get the representative relative value of stable contacts:

$$R_{sc} = \frac{N_{sc}}{N_{allc}}.$$

- *Stale message ratio* (micro metric) further describes the basic metric ‘number of stale messages’. Ratio R_{sm} represents the share of the messages sent to the offline peers ($\sum N_{lm}$) in the overall number of sent messages ($\sum N_{allm}$):

$$R_{sm} = \frac{\sum N_{lm}}{\sum N_{allm}}.$$

- *Recovery time* (macro metric) describes the time that an overlay needs to regain the value $\text{Perf}_{\text{ref}_2}$ of an observed metric from a current value $\text{Perf}_{\text{ref}_1}$, caused by overlay failures or perturbations. If Perf_{t_1} and Perf_{t_2} are the values of the observed metric in the time points, t_1 and t_2 , we define the recovery time as:

$$\text{if}(\text{Perf}_{t_1} = \text{Perf}_{\text{ref}_1} \wedge \text{Perf}_{t_2} = \text{Perf}_{\text{ref}_2}) \Rightarrow t_{\text{recovery}} = t_2 - t_1.$$

The presented metrics are used to evaluate the quality aspects of the overlay and identify the origin of possible poor performance (e.g. if response time is low, and the number of stale messages is high, it might be that the failure detection is not timely).

Workloads consist of performed queries, churn, and geo-distribution of peers

5.3 WORKLOAD

In order to evaluate the quality of a system, appropriate system operations, its frequency and intensity, user behavior, and affecting environmental factors (*workload*) must be set to the system. The workload for the peer-to-peer overlays consists of joining, query operations, leaving, and failing of the peers. The turnover of the peers (leaving/failing and rejoining) is referred to as churn. This is one of the most studied and analyzed components of user behavior in peer-to-peer systems. The impact of the churn on the quality of the overlay performance is significant [PKL⁺08]. Additionally, part of the workload especially important for evaluation of Globase.KOM is the geographical distribution of peers on the map and geolocation-dependent churn. Summarized, defining workloads here means setting the operations peers perform and appropriate time spans (Section 5.3.1), their geographical distribution (Appendix D), and churn rates (Section 5.3.2).

5.3.1 Experiment Timelines

Different experiment scenarios to address all quality aspects

Experiment timelines are the timespans for the overlay operations, changes in environmental factors (e.g. churn, unstable network conditions), and measurements. Operations performed by the peers are further specified as *local* or *distant*, depending on the distance between an initiator and the geolocation of the queried point for lookup and find the closest or the center of the queried area in area search. For local operations, this distance is zero or the query results are inside the responsibility zone of the initiator's parent superpeer. The three main experiment timelines are used:

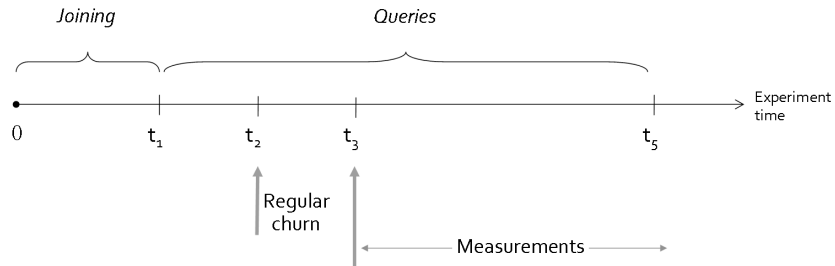


Figure 37: Efficiency experiment timeline

- *Efficiency experiment timeline:*

This timeline is used as the basis for many experiments. The main aim of these experiments is to find the optimal workloads whereby the overlay performs the best. Additionally, it should capture the overlay behavior under regular, moderate churn (explained further in the next Section 5.3.2). It is also important to consider the real user behavior and examine how the overlay handles it.

In Figure 37 we depict the efficiency experiment timeline. All N peers join the overlay network in the first t_1 minutes. They

then begin periodically to perform local and distant lookups, area searches and find the closest. After the overlay network stabilizes, at the timepoint t_2 , the regular churn accompanies the periodical queries. After stabilization and after all scheduled operations take place, the measurement starts (time point t_3).

- *Stability experiment timeline:*

These experiments addresses the quality of the overlay under intensive queries and leaving (but not failing) of a high number of peers in a short time span.

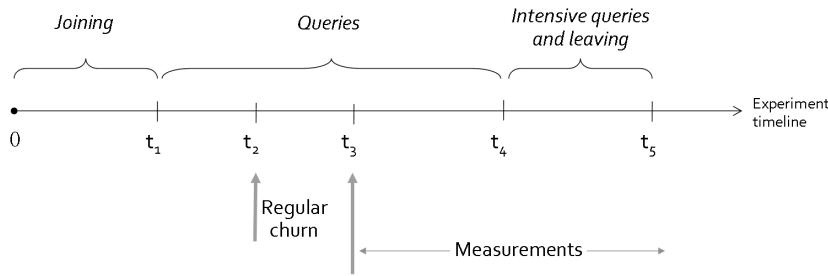


Figure 38: Stability experiment timeline

The timeline is based on the efficiency experiment timeline with the difference being in timepoint t_4 (see Figure 38) where intensive queries and leaving start. The difference in the quality of the overlay is therefore captured in the time span between t_3 and t_5 .

- *Robustness experiment timeline:*

The main goal of these experiments is to examine the overlay network behavior under extreme churn rates and critical failures (e.g. when a particular set of superpeers, those crucial for the routing, simultaneously fail). In that way, the maintenance strategy, in particular, is addressed.

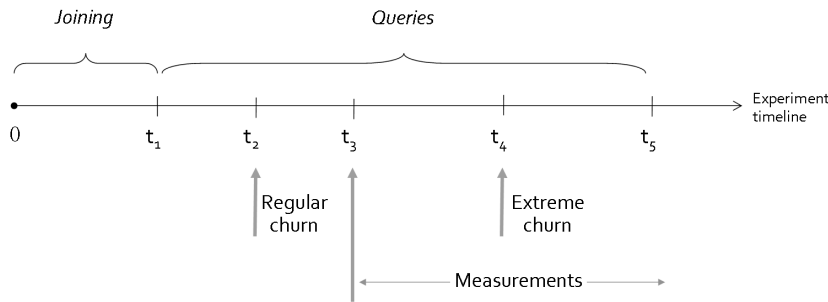


Figure 39: Robustness experiment timeline

This timeline is depicted in Figure 39. After the stabilization phase, under the regular churn (time point t_3), periodical queries continue. However, at the timepoint t_4 , either extreme churn starts, or the crucial superpeers begin simultaneously failing (e.g. connected superpeers in the same branch, or the root superpeer).

5.3.2 Churn Models

How long are the peers online, when do they join, fail, etc

Churn is one of the most studied and analyzed components of user behavior in peer-to-peer systems. The impact of churn on the quality of the overlay performance is described in [PKL⁺08].

During a peers' participation in the system over its *lifetime*, a peer repeatedly connects and disconnects from the system, dividing its participation into *sessions*. An example of a user participation in a peer-to-peer overlay network is presented in Figure 40.

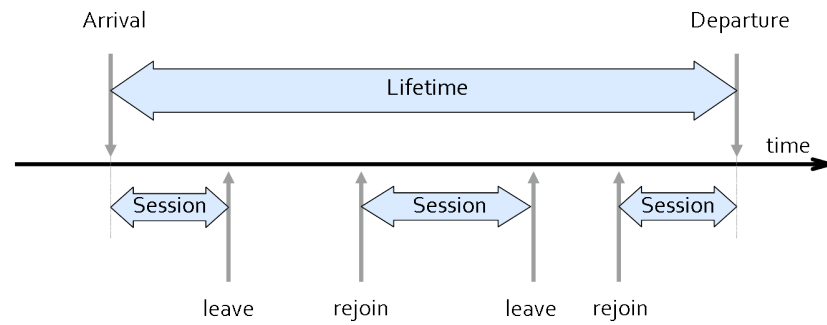


Figure 40: The lifetime of a peer consisting of several session intervals [PKL⁺08]

The majority of existing churn models are based on measurements on popular file-sharing applications like Napster, Kazaa, FastTrack Gnutella, KAD, and BitTorrent. Depending on the subject of the modeling, these models can be divided into lifetime and session models. The first class of models describe when a peer joins the network for the first time and leaves for the last time. These models consist of *arrival process* models such as deterministic (where peers arrive with a constant rate), Poisson (where peers arrive independently from each other, with the exponential arrival rate) or *lifetime distribution* (e.g. exponential, Pareto). The second class of models focus on the peers' online and offline session times, during their lifetime. These models observe the churn on three levels – *global*, *group*, and *peer*, and define the session and intersession durations using e.g. exponential, Weibull, or log-normal distribution. In [SR06] Stutzbach and Rejaie describe the online behavior of users in three applications – Gnutella, BitTorrent, and KAD, and show the similarities in the participation dynamics across all three systems. Additionally, they proved that a large portion of peers have long session times, which are not exponential, and the correlation between the session time lengths of peers which joined consecutively.

Variety of existing churn models based on file-sharing applications

Lifetime churn models are relevant for content distribution applications. A content distribution overlay network (so called swarms in the case of BitTorrent) is created for the distribution of a single file and as soon as users download the file, they leave the system for good. In applications like VoIP, multimedia streaming, and some file-sharing applications' lifetime is significantly longer than the session times. That makes lifetime of the peers a much less influencing factor on the quality of a system. Application scenarios of Globase.KOM offer similar

Lifetime of a Globase.KOM peer is long \Rightarrow Session time churn is of more of interest to us

incentives to these applications for users to participate in the system for a long time. It is an interactive application, as a user benefits from the fact that other users use its services. Therefore, session time churn models are of more interest to us.

The churn model used for our workloads needs to distinguish online behavior of superpeers and regular peers. We used, therefore, the model described by [HT07] as the basis. Three types of peers are included in this model: Benefactors (Be) with long session times, Peers (Pr) with regular session times, and Peepers (Pp) with short session times. For each of these three groups, a connectivity factor $CF = \frac{ON}{ON+OFF}$ is defined, where ON and OFF denote the lengths of online and offline session times respectively. An average length of online session times is assigned to each of the groups separately. Globally, a churn factor CH is defined as the average portion of the the overlay network population N that will be offline during the experiment. In our experiments, we used power law distribution (as it best models Skype user behavior, detailed in Appendix D).

As the geographical location of the participating peers affects the structure of the Globase.KOM overlay, the relation between the online behavior of the users and their location is of special interest. We have conducted an experimental measurements on Skype, peer-to-peer VoIP application in order to capture the correlation between churn, geographical location, and distribution of the peers on the map. The Skype online user behavior and geographical location is used as input for experiments. Further details on these experiments and captured behavior is presented in Appendix D.

We observe Skype user behavior, and captured correlation between churn and geolocation

5.4 MEASURING QUALITY PROPERTIES

As mentioned in the beginning of this chapter, it is important to define the methodology for assessing the non-functional requirements: efficiency, scalability, fairness, stability and robustness. In order to do so, for each quality property an appropriate combination of metrics and workloads needs to be defined. Based on the definitions given in Section 1.2.3, we specify experiment timelines, churn models, queries to be performed and metrics.

Assessing the quality properties means designing the experiments accordingly, in terms of workload and metrics

- *Efficiency*

As the efficiency of an overlay shows the ratio of performance and costs, we used retrievability and response time as performance metrics and the corresponding costs metrics are the number of overlay hops and relative delay penalty. We used the efficiency experiment timeline (see Section 5.3.1) together with the geographical distribution and location-aware model, presented in the previous section (Section 5.3).

- *Fairness*

Fairness means that the costs for overlay operations are uniformly distributed over the peers, proportionally to their individual capacity. We distinguish fairness (or load-balancing) among peers

and superpeers, separately because the load of a peer should be proportional to its individual capacity. We observe fairness in all three experiment timelines, using the load balance ratio (LBR) as the metric for fairness.

- *Stability and Robustness*

The stability of an overlay addresses the variation of quality under intensive overlay operations (queries, leaving, joining), while robustness describes the variations under critical failures and massive churn rates. We use the stability and robustness experiment timeline for this evaluations. Variation of quality is measured by the size of the peak and the recovery time (as presented in Figure 41). The measured metrics are the same as for efficiency. Further metrics, stale message ratio and stale contact

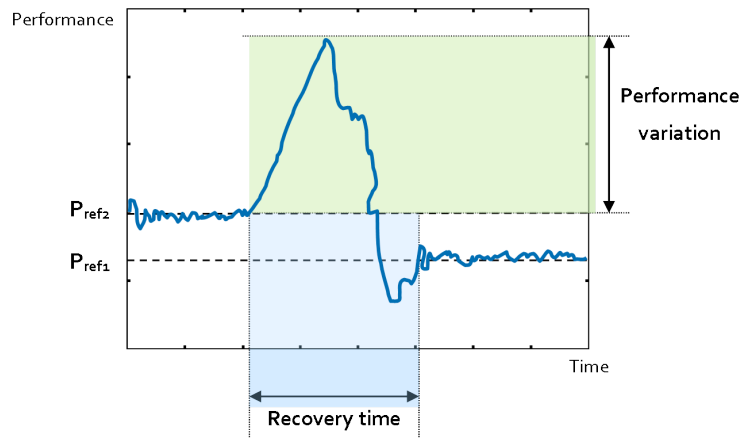


Figure 41: Measuring recovery and variation for stability and robustness

ratio, are used to further identify the overlay design aspect that led to long recovery times or peaks in the observed metric.

- *Scalability*

Scalability should show that the size of the network does not greatly influence performance or costs. We, therefore, used the efficiency timeline and all efficiency metrics and observed their behavior while increasing the number of participants.

- *Validity*

Validity describes the completeness and correctness of the query results. The main metric for validity is certainly retrievability. All experiment timelines are relevant for this quality property, are the same as these for efficiency, stability and robustness, as the validity partly describes functional requirements.

Observing stale message ratio and stale contact ratio we can identify the reasons for invalid query results.

5.5 EVALUATION TECHNIQUES

Evaluation techniques for peer-to-peer overlay networks include using an analytical model, testing prototypes in testbeds, or involving simulation. As peer-to-peer systems are very complex, analytical approaches typically lead to too many simplifications. Running large scale experiments in a testbed with prototypes is difficult due to a lack of sufficiently large testbeds. Only PlanetLab [Pla06] is a possible alternative as a testbed with about 1035 nodes at 494 sites (May 2009). However, it is still not sufficiently large [SPBP06] to provide a precise snapshot of a p2p system with its millions of participants. An additional reason why testbeds are not appropriate for our evaluations is that there is no possibility of placing the peers according to the captured geographical distribution. The approximations which simulations provide are much closer to reality than an analytical approach, it is possible to simulate networks of hundred thousands of peers, and to vary the geolocation of the participants according to the needs of various workloads.

In [KKM⁺07] we showed the requirements for a simulator of peer-to-peer systems. There are: modularity, realistic and flexible underlay network and user behavior models, the possibility for modeling resources and services, easy experiment setup and scalability. We gave an overview of the existing simulators for peer-to-peer overlays and systems and identified the need for a general simulation framework that fulfills all stated requirements. The evaluation of Globase.KOM requires in particular, the support for geographical dependent peer distribution and churn rates. We give an overview of the most important features of the proposed simulation framework, namely PeerfactSim.KOM.

PeerfactSim.KOM [Pee] is a discrete-event based simulator, written in Java, with a modular designed. It consists of six layers presented in Figure 42. These are identified as the key components of the widely deployed peer-to-peer systems based on the analysis of their functionality and supported services. Each layer encapsulates important aspects in order to model a peer-to-peer system in its entirety.

Regarding the location-dependent features, it supports the distribution of the peers according to the predefined density world map (*bitmap-based random* distribution). Additionally, location-aware churn can be simulated as well. In a simulator, the virtual space where the peers are located is represented by an Euclidean plane. Each joining peer obtains a unique two-dimensional coordinate according to its position. The distribution of peers on this plane significantly affects the simulation. Two variants for modeling a peer distribution, which are important for location-aware churn model (see Appendix D), have been realized:

- *Uniform random*: According to this distribution, peers in the network are distributed uniform-randomly on the Euclidean plane. The advantage of this distribution is its simplicity. However, in order to capture the effects of distribution on the overlay, this model is not able to give the realistic picture.

Simulation is chosen because it is very flexible and supports experiments on a large scale

PeerfactSim.KOM allows large-scale simulation of peer-to-peer systems in their entirety

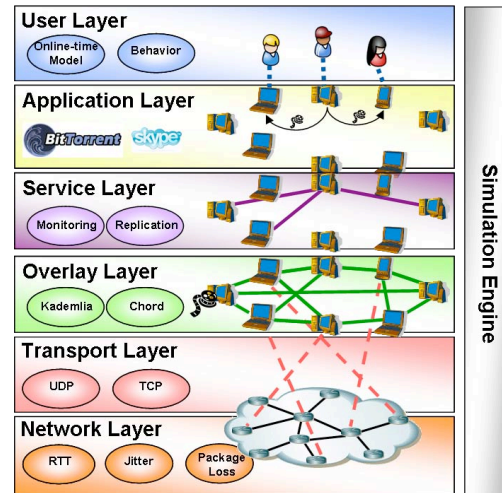


Figure 42: Functionality layers of PeerfactSim.KOM

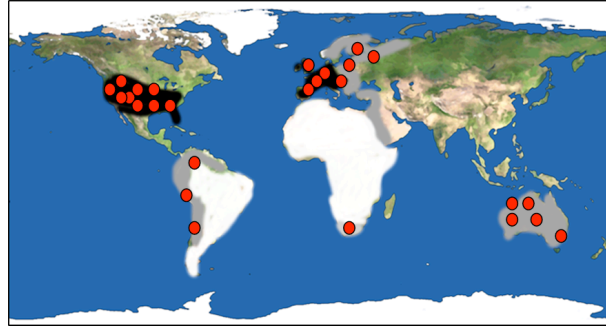
- *Bitmap-based random*: The peers' distribution is not uniform across the world as peers form clusters concentrated at certain parts of the world whilst other vast areas are deserted. In order to simulate such non-uniform distribution, peers are randomly distributed based on a grayscale colored bitmap. This bitmap can be a map of the world or a map of a smaller area, like the map of a city. Sparsely populated areas are represented by a lighter gray and darker areas represent the denser areas. Therefore, the darker an area is on the bitmap, the higher the probability that there is a peer mapped at this location (see Figure 43). In our simulations, we used a density map of captured Skype superpeers, depicted in Figure 92.

5.6 SUMMARY

This chapter provides the answers to the questions that are necessary to be addressed in order to carry out the evaluation correctly, which are stated at the very beginning of this chapter:

- The goal of our evaluations is to prove that our solution (Globase.KOM) fulfills the functional and non-functional requirements. Additionally, the influence of the system-wide parameters of Globase.KOM on the quality of the solution needs to be examined.
- Metrics that we will be using are divided into basic (e.g. response time, number of overlay hops) and derived (e.g. relative delay penalty, load balance ratio).
- We specify how to address the non-functional requirements (efficiency, scalability, stability, robustness, validity) in experiments and evaluated these quality aspects of our solution.

Answers to the questions from the beginning of the chapter



(a) Grayscale colored map of the world with the generated peers



(b) Grayscale colored map of the center of Darmstadt



(c) Resulting grayscale-bitmap and peer distribution

Figure 43: Geographical maps colored with grayscale reflecting the concentration of users - the darker parts represent areas with more users relative to lighter parts on the map

- Various workloads for the solution in experiments are defined for each evaluation goal. They are presented in the form of experiment timelines and churn models. We discuss churn models in detail and observe the online behavior and geographical distribution of Skype users. A popular peer-to-peer VoIP application Skype can represent the realistic user behavior for Globase.KOM due to its interactive nature (in comparison with file-sharing systems, where users go offline as soon as they download the requested file).
- The system wide parameters of Globase.KOM are load thresholds L_1 and L_2 , the number of interconnections, frequency of all types of failure detection mechanisms used, and query timeouts. They need to be set to the optimal values according to the application scenarios.
- Simulation is chosen as evaluation technique, because of its flexibility and scalability. The used PeerfactSim.KOM, a simulation framework for peer-to-peer systems. We described the Globase.KOM simulation model.

The next chapter presents the evaluation results, based on the methodology discussed in this chapter, regarding parameter calibration, the assessment of quality aspects, and comparisons to the related work.

After the overlay design, the mechanisms and algorithms of our solution are described; the evaluation of the requirements stated in Section 1.2 is a necessary step to prove the viability and efficiency. The previous chapter (Chapter 5) presents the evaluation goals and appropriate methodology to address them.

This chapter presents the evaluation results of our solution, Globase.KOM, which corresponds the stated goals and methodology. We, therefore, first we present the calibration of the system-wide parameters in Section 6.1. We discuss the influence of the load thresholds parameters L_1 and L_2 on the shape of the tree and quality of overlay operations and analyze the chosen values. In order to set the number of interconnections, a trade off between efficiency and load balance on one side and the costs of maintenance overhead on the other side must be taken into account. The size of cache, timeouts of queries, and parameters for failure detection mechanisms have influence on robustness and retrievability of the overlay and can also introduce additional maintenance costs. At the end, we summarize and analyze our choice of parameter values.

We explain the parameter calibration, assess quality aspects, and compare with related work

Sections 6.2 - 6.6 address the non-functional requirements: efficiency, fairness, scalability, robustness, and stability. Functional requirements are addressed in each of this sections by evaluating the retrievability of the queries under various workloads. The comparison with the related solution RectNet, is presented in the Section 6.7.

Finally, Section 6.8 provides the answer to whether our solution fulfills the requirements stated in Section 1.2; and Section 6.9 summarizes the obtained evaluation results.

6.1 PARAMETERS CALIBRATION

Globase.KOM has several variable system-wide parameters that influence the quality of overlay. They are listed in Section 5.1: load threshold parameters L_1 and L_2 , the number of interconnections, size of cache, timeouts of operations, frequency of periodical failure detection messages, and the number of missing periodical liveness information before a failure is detected. The load threshold parameters L_1 and L_2 have the biggest influence on the structure of the overlay, routing and therefore the quality of overlay. We will discuss it in the separate Section 6.1.1. Interconnections are crucial in improving the efficiency of routing, taking the load from the superpeers in the higher levels of the tree. However, with the increasing number of the connections each peer has to maintain, the protocol overhead rises. This is discussed in Section 6.1.2. Both load thresholds and interconnections affect the overlay structure and are specific to our solution. Parameters in failure detection mechanisms, size of the cache and operation timeouts are general parameters for

Why is the parameter calibration challenging?

Protocol parameters	Value(s) (a)	Value(s) (b)
Ratio $\nu = \frac{L_1}{L_2}$	0.5	0.3, 0.4, 0.5, 0.6
Load threshold L_2	10, 20, ... 100	60, 70, 80
Number of interconnections S_1	20	
Size of cache S_2	10	
Timeout for operations T_1, T_2	2 sec.	
Simulator settings	Value(s)	
Number of peers	1 000 (5 000, 10 000)	
Experiment timeline	Efficiency timeline	
Churn	None	

Table 1: Parameters for evaluation of effects of load threshold L_1 (a) and ratio ν (b) on the shape and size of the tree

each overlay and are dependent on the underlay network condition and user behavior. Their settings will be presented in Section 6.1.3.

6.1.1 Load Threshold Parameters

What are parameters L_1 and L_2 and how can they influence quality?

In our solution, a superpeer (publicly reachable, static peers with more capacity, spare bandwidth, and good network connectivity) is responsible for all peers in its zone. These rectangular zones are formed using a clustering algorithm for defining highly loaded areas (see Section 4.5.1) and are assigned to one peer in that area, which becomes the superpeer of a zone. As metric for the load of an area we use the number of peers in the zone (see Section 4.5). There are three load levels - *normal* (below a threshold L_1), *overloaded* (between thresholds L_1 and L_2), and *critically overloaded* (above L_2). Once a superpeer's load exceeds the threshold L_2 , it runs the clustering algorithm and gives away approximately $L_2 - L_1$ peers. As the neighbor relations in the tree structure are a direct reflection of the number of zones and their relations (e.g. intersections), the values of L_1 and L_2 have a strong influence on the size and the shape of the superpeer tree.

First we examine the effects of threshold L_2 and then the ratio $\nu = \frac{L_1}{L_2}$. All simulations in both cases were done with protocol parameters and simulator settings according to Table 1 for 1 000 peers. Detailed settings for simulations with 1 000, 5 000 and 10 000 peers are provided in Appendix F.

Effects of Load Threshold L_2

In the simulations with 1 000 peers, the values for L_2 were set from 10 to 100, while L_1 was fixed at half the size of L_2 . When the load threshold L_2 is very low (relative to the number of participants in overlay) a large number of very small zones is created while the network builds up. That implies a broader tree, where a root superpeer has a number

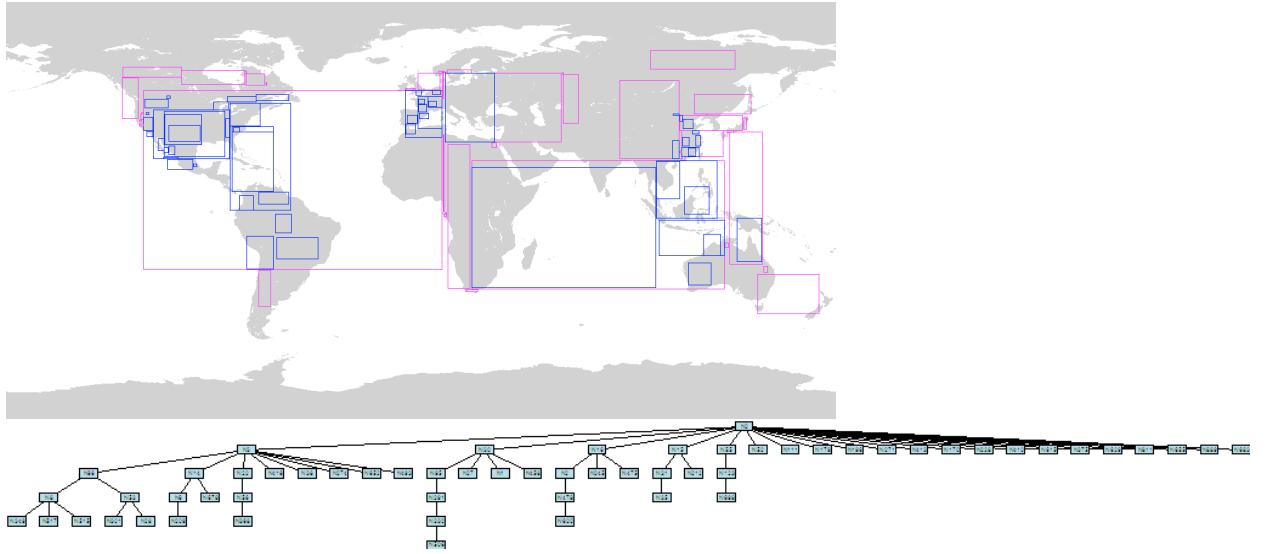


Figure 44: Zones and corresponding basic superpeer tree for low $L_2 = 30$

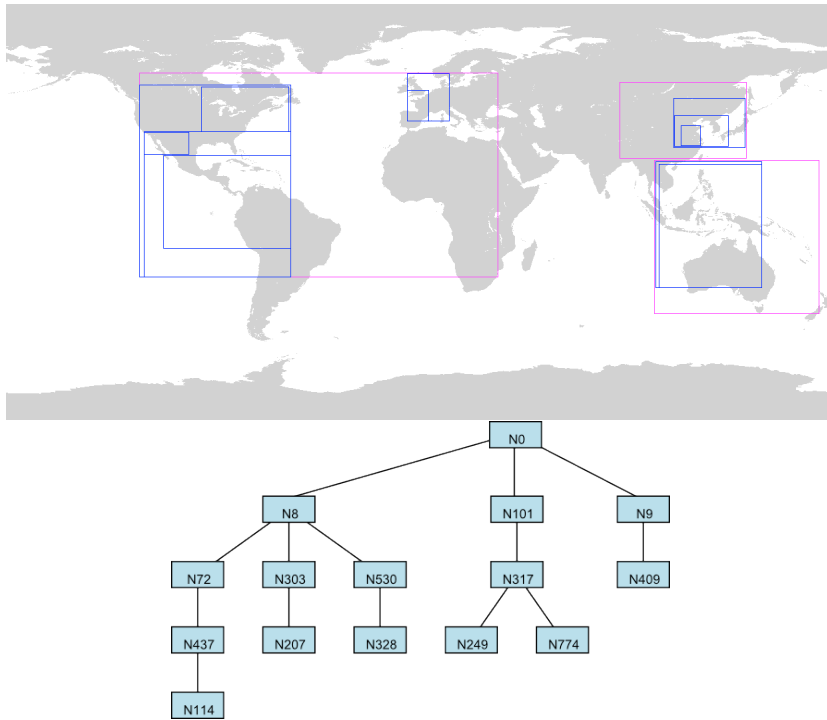
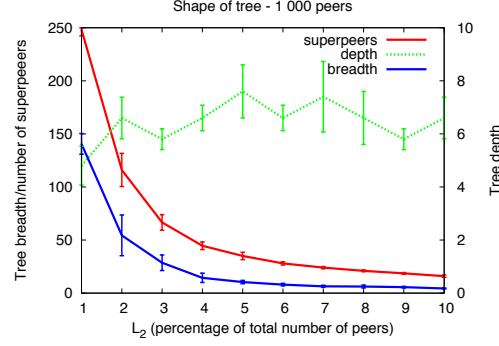


Figure 45: Zones and corresponding basic superpeer tree for high $L_2 = 100$

Figure 46: Shape of superpeer tree for different L_2 and 1 000 peers

of direct children superpeers. For example, a tree with $L_2 = 10$ has approximately 250 superpeers, a breadth of 130 and depth of 4. Because of its extreme dimensions, we present zones and corresponding tree with $L_2 = 30$ instead of $L_2 = 10$ in the Figure 44, as an example tree for low L_2 .

In a tree with $L_2 = 10$, each superpeer is responsible for the most $L_2 - L_1 = 5$ peers and the root superpeer is additionally responsible for 130 children superpeers. A ratio of peers to superpeers is just 4 to 1. Such a tree shape implies a massive load for the root superpeer (query messages and maintenance overhead) and deviate greatly from the peer-to-peer communication paradigm. We can, however, expect low number of hops needed to resolve the queries and low response time on the cost of having the root superpeer as the bottleneck of the communication.

Figure 45 shows the zones and corresponding tree with higher $L_2 = 100$. Choosing a higher value for L_2 means that more peers are needed before the new inner zone is formed. Zones are consequently larger than the zones in Figure 44. Fewer zones will be formed in the first level of a tree hierarchy (direct children of root superpeer) which results in smaller breadth of a tree. Such a tree has on average, 20 superpeers, a breadth of 3, and depth of 4. Each superpeer is responsible for $L_2 - L_1 = 50$ peers and the root superpeer is additionally responsible for 3 children superpeers thus the ratio of peers to superpeers rises to 50 to 1.

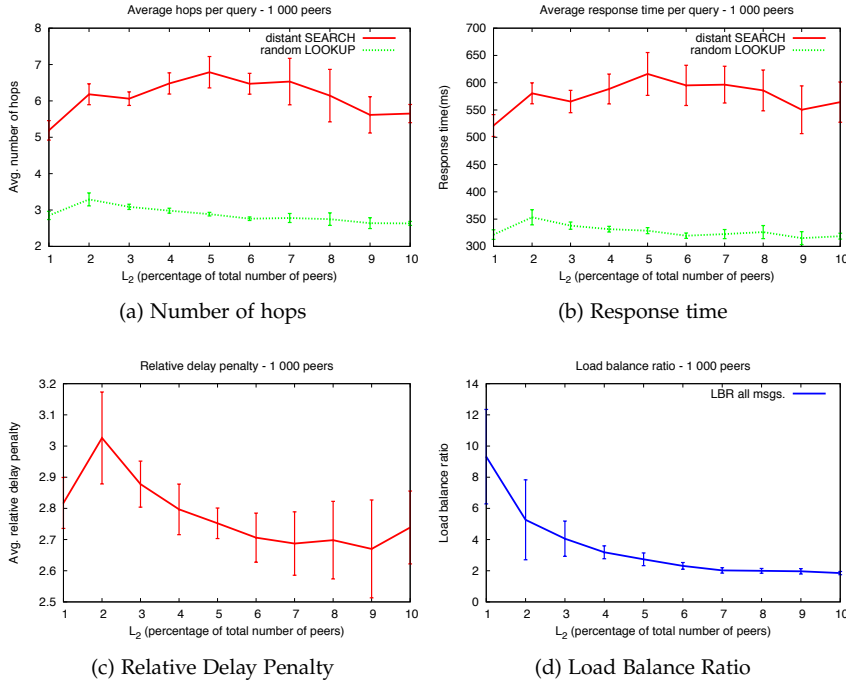
The depth of the tree in both cases is the same (4) which implies similar response time and number of hops needed to resolve the queries. However, in the case of higher L_2 , a load between superpeers is significantly better balanced.

Figure 46 shows the number of superpeers and the depth and the width of the tree for all L_2 values investigated in the simulations for 1 000 peers (values are given with 95% confidence intervals over the the simulation runs).

As previously discussed, low L_2 values result in an largely increased number of superpeers in the overlay. The depth of the superpeer tree is not affected by changing the load threshold L_2 . Average values for the

Low values of L_2
make a tree very
broad and place
massive load on the
root superpeer

Higher values of L_2
make a tree well
balanced

Figure 47: Effects of L_2 on query performance and costs

tree depth were measured between 5 and 7 for L_2 . The breadth of the tree, on the other hand, is highly effected by load threshold L_2 . Low values for L_2 result in a very broad tree in the first level, resulting in huge number of children superpeers of root superpeer. Results of our simulations for 5 000 and 10 000 follow the same behavior.

In the following we present the effects of L_2 on the performance of *lookup* and *distant area search* and on network costs. Figure 47 shows the influence of the size of L_2 on the hops needed per average query, the response time and the load balance ratio of Globase.KOM. One reason for the smaller number of hops and response time for low L_2 is a very broad tree with a high number of root children. However, the extreme breadth of the tree also reflects in an excessively high load balance ratio for L_2 values smaller than 4% of the network size. With rising values for L_2 , a load balance ratio of around 2 can be achieved. Except for the aforementioned low values, the number of hops and response time of both *lookup* and *distant area search* queries are not significantly affected by changing the load threshold L_2 . The relative delay penalty of Globase.KOM is not strongly affected by changing the L_2 value. The average relative delay penalty varies between 2,6 and 3,1, decreasing slightly for higher L_2 values. The low relative delay penalty for the load threshold L_2 set to only 10 (which is 0,1% of the network size) is once again caused by the low number of hops due to the extreme breadth of the superpeer tree.

We can see that the value of the load threshold L_2 has a big influence on the breadth, while hardly any effect on the depth of the tree. Low

Load balance ratio and relative delay penalty are the most affected by L_2

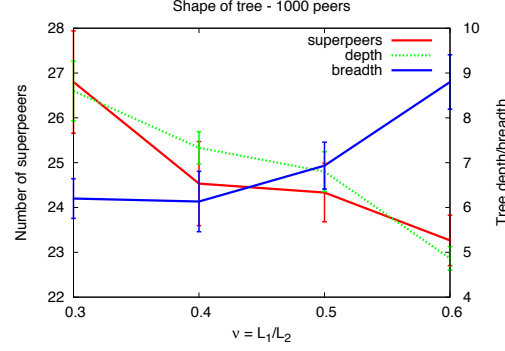


Figure 48: Shape of superpeer tree for different ratio $v = \frac{L_1}{L_2}$

A review of the influence of L_2 on tree shape and quality

values of L_2 result in small zones and a tree with a huge number of superpeers. The number of hops and response times of queries change insignificantly with the varying L_2 . However, the biggest influence of L_2 is on the load balance ratio, where the root superpeer receives from 2 to 10 times more messages than the median loaded peer. Bigger values on L_2 reflect better load distribution. However, the optimal L_2 depends strongly on the size of the network and scenario of usage and will be discussed in summary of this section.

Effects of Ratio $\frac{L_1}{L_2}$

In the previous evaluations we chose L_1 to be half of the value of L_2 . The influence of the ratio $v = \frac{L_1}{L_2}$ on the shape and size of the superpeer tree needs to be evaluated separately. When a new zone is created inside the existing zone of an overloaded superpeer, L_1 determines the number of peers that are taken from the overloaded superpeer and passed to a newly assigned child superpeer, namely $L_2 - L_1$. The larger the number of the peers in the newly formed zone is, the sooner the threshold L_2 in the newly formed zone will be reached. At the same time, the parent zone, from which this large amount of load is taken by forming the new zone, is now responsible for very few peers. This means that we can expect a deeper tree when ratio v is smaller.

We can expect a deeper tree when the L_1 value is significantly smaller than that of L_2

The impact of v on the shape and size of the superpeer tree is shown in Figure 48. We can see that the effect of ratio v on the size and breadth of the tree is not as strong as the influence of L_2 . As L_1 denotes the number of peers that remain in the zone of responsibility of a superpeer after creating a new inner zone, the number of superpeers in the network and the depth of the tree decrease slightly for larger L_1 values, i.e. bigger ratio v . For the same reason effect of v on the breadth of the superpeer tree can be observed. With higher L_1 values or larger ratio v , the overloaded superpeer remains responsible for a larger number of peers. This makes further splitting operations for an once overloaded superpeer more likely, resulting in a broader tree. This characteristic can be seen when comparing the two trees in Figures 49 and 50.

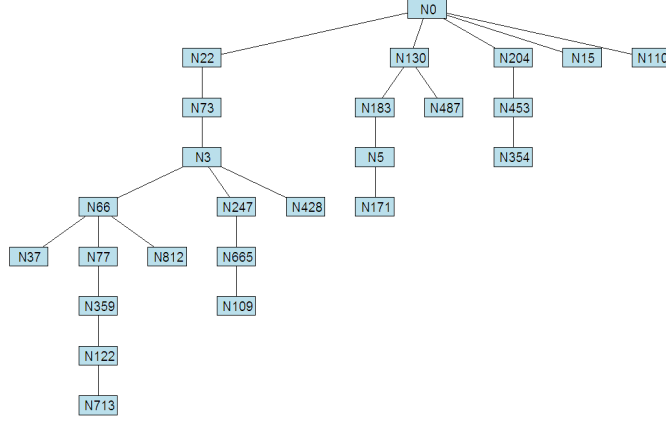
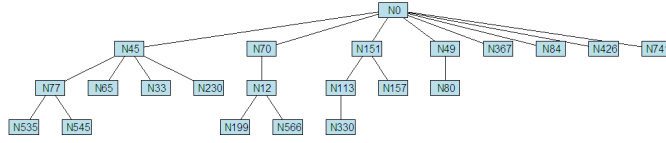
Figure 49: Superpeer tree for low ratio $\frac{L_1}{L_2}$ ($L_2 = 70$ and $L_1 = 21$)Figure 50: Superpeer tree for high ratio $\frac{L_1}{L_2}$ ($L_2 = 70$ and $L_1 = 42$)

Figure 51 shows the effects of ν ratio on query performance (response time) and costs (average number of hops, relative delay penalty, and load balance ratio). The average number of hops for *lookup* and *distant area search* benefits from the broader tree caused by larger values of ν . While the number of hops for *lookup* only decreases slightly for larger L_1 values, the hops for *Distant Area Search* falls from an average of over 7.5 (for the ν of 0.3) to just under 5.0 for a ν of 0.6. The downside of broad trees, poor load distribution among the superpeers, can also be observed in the results of the simulation runs for larger ν values. The load balance ratio for Globase.KOM increases by 30% when reaching a ν of 0.6 (in comparison to the average load balance ratio of lower ν values). Taking into account that superpeers have better connectivity and bigger capacities, a load balance ratio of both 2.0 and 2.6 is a good value. However, this value will be further decreased by interconnections (see Section 6.1.2)

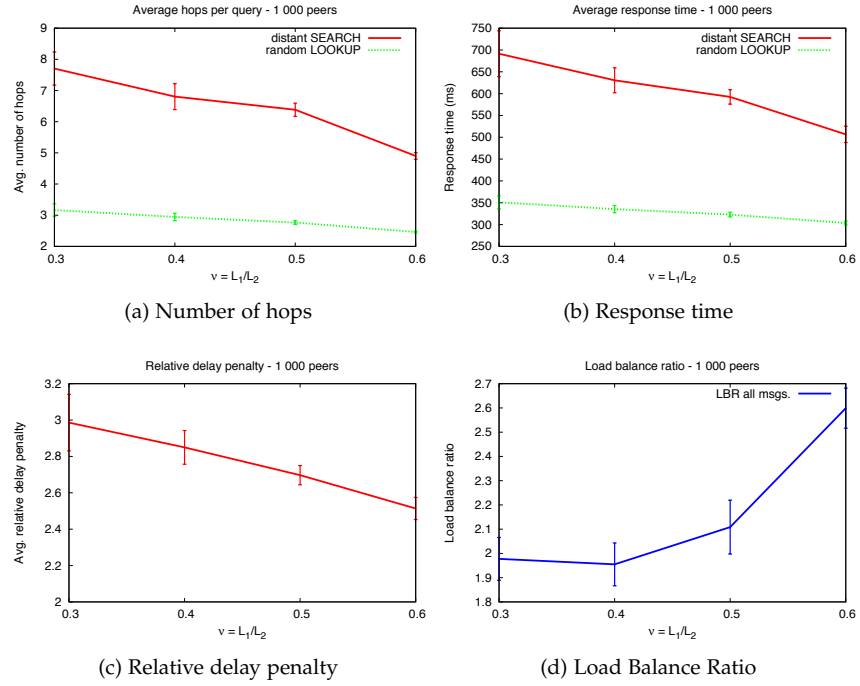
The response times for queries show similar dependencies on load threshold L_1 . The response time of *lookup* can be reduced by 8%, if adequate values for L_1 are set. *Distant Area Search* can be accelerated by 15%, before the increase in load balance ratio becomes extensive.

The relative delay penalty also benefits from the change of the shape of the superpeer tree for larger L_1 values. The average relative delay penalty decreases from just under 3 for a $\nu = 0.3$ to an average of 2.5 for a $\nu = 0.6$. A relative delay penalty improvement of 10% can be achieved before the extensive increase of the load balance ratio occurs.

Based on this evaluation and analysis, we set the ratio $\nu = 0.5$ in the further evaluations and in the prototype.

All metrics are
affected by choice of
 $\frac{L_1}{L_2}$

We set $\frac{L_1}{L_2}$ on 0.5

Figure 51: Effects of v ratio on query performance and costs

Summary

Setting the L_2 value relative to network size or as the absolute number?

As we can see, the shape of the tree has significant impact on fair distribution of the load among the superpeers. A well balanced tree gives good performance and costs values. The value of load threshold L_2 effect the breadth, while the ratio $\frac{L_1}{L_2}$ influences on the depth of the tree. We chose 0,5 as a good value for the ratio $\frac{L_1}{L_2}$, which has proven to create good performance results together with a proper load distribution among the superpeers. Setting a good value for the load threshold L_2 depends on the size of the network and placement of the nodes. We saw that bigger values of L_2 , in relation to the size of the network, result in a better load balance. That, however, rises numerous questions: How do we know what the size of the network will be, and do we need to reform the existing zones accordingly? Does it mean that for one million peers, a superpeer should be responsible for 80 thousand peers in order to have fair distribution of the load among superpeers? Taking these issues into account, we have the following possibilities for choosing the value L_2 :

Setting an absolute L_2 value, without adding massive load to the root superpeer

1. A special algorithm for forming the zones of the root superpeer makes sure that not more than e.g. 5 zones are made inside of the zone of the root superpeer. That means that the zones of direct child superpeers will be made intentionally bigger than the recognized hotspot itself. That can be achieved by using the rough map of possible positions of peers (e.g. using the map of Skype or Internet users). Values of L_1 and L_2 should be set to 50 and

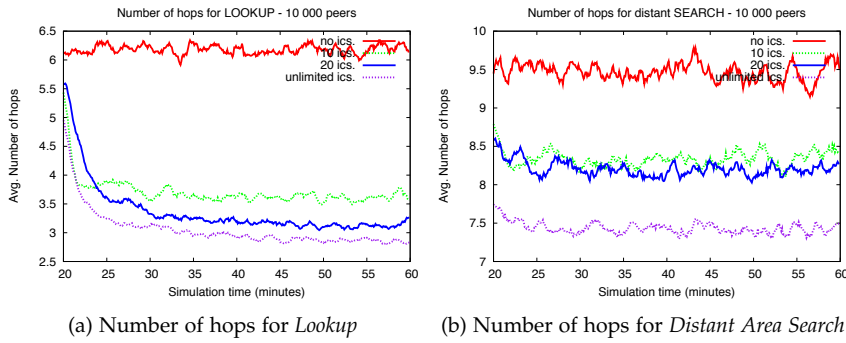


Figure 52: Effects of number of interconnection on number of hops for queries

100 (similar settings as in popular superpeer-based peer-to-peer applications Skype [Skyb] and Limewire [Lim]).

2. Direct child superpeers of the root superpeer must have interconnections to each other. Additionally, the failure recovery mechanism would use the connections to these superpeers. In that way, a number of the messages for resolving the queries and failure recovery a root superpeer receives would be significantly reduced.

We chose L_2 to be between 100 and 120, and used both optimizations. In that way, the breadth of the tree is not more than the set limits (we chose 5) and a superpeer is responsible for the reasonable number of peers in its zone.

6.1.2 Number of Interconnections

Interconnections are crucial in improving the efficiency of routing by taking the load from the superpeers in the higher levels of the tree. At the same time, the increasing number of neighbor connections increases the overhead. Additionally, as interconnections should connect superpeers from different branches of the tree, their physical, geographical distance is higher than in the case of child/parent connections. Frequent usage and maintenance of interconnections can therefore greatly influence the relative delay penalty. To evaluate the effects of the number of interconnections on the performance and costs of overlay, simulations were done with 10 000 peers and different settings for the number of interconnections (0, 10, 20, no limitation). Since *local area search* and *find closest* only have a regional routing radius, the evaluation of interconnection effects focuses on *distant area search* and random *lookups*. The simulation was done with protocol parameters and simulator settings as described in the Table in Appendix 20.

Figure 52a shows the effect of the interconnections on the *lookup* operation. Without the use of interconnections in the superpeer tree, the number of hops needed for an average *lookup* operation stays over 6 hops for the entire time of the simulation. When interconnections are enabled in the superpeer tree, they show an immediate effect on

What are interconnections and how can they influence quality?

The more interconnections, the smaller the hop count and response time for lookup query

the number of hops for *lookup* operations. When allowing just 10 interconnections per superpeer, the number of hops used by *lookup* operations decreases from the beginning of the simulation. After the first interconnections are being built during the joining phase of the scenario (not shown in Figures 52a and 52b), their effect on the *lookup* operation increases when the user actions in the scenario start to begin after 20 minutes of simulation time. After another five minutes (which corresponds to only 4 user operations per peer), the average number of hops using 10 interconnections per superpeer decreases to less than 4 hops (which is 35% better than without interconnections) and stays at this level throughout the rest of the simulation.

Allowing more interconnections within the superpeer tree further reduces the number of hops need for *lookup* operations. When using up to 20 interconnections the average number of hops can be reduced to 3.3 (which is 46% less than without interconnections) for *lookup*. Permitting an unlimited number of interconnections between superpeers does not lead to an excessive further reduction of the number of hops, but introduces some negative effects on the load distribution within the superpeer network, as this evaluation will show later on (Figure 54b). The hop count for unlimited interconnections lies just below 3.0 hops per *lookup* (saving around 50% of the hops in comparison to not using interconnections).

The performance of area search also benefits from interconnections

The effect of enabling interconnections in the superpeer tree on the *Distant Area Search* operation, shown in Figure 52b, is not as significant as for *lookup* operations, but it is definitely noticeable. Interconnections can only speed up routing the *search* query to a superpeer whose zone of responsibility contains or intersects the search area. Further forwarding to parent or child superpeers, which is in most cases inevitable for a complete query resolution, cannot be influenced by the interconnections. Average values for the number of hops for *area search* vary around 9.5 hops throughout the simulation when interconnections are not being used. Allowing up to 10 interconnections in the superpeer tree reduces the number of hops needed for an average *search* operation by 12%. Setting the number of interconnections to 20 decreases the number of hops slightly further to 8,3 hops. A reduction of more than 20% can be achieved by not limiting the number of interconnections. Figure 53a shows the effects of interconnections on the response time of *lookup* queries. The response time for *lookup* lies between 550 and 600 ms throughout the simulation, if no interconnections are used. Similar to the development of the number of hops, the average response time drops below 400 ms, if the usage of interconnections is enabled. 10 interconnections decrease the average time for responding to a *lookup* query by 35%. An average response time of less than 350 ms can be achieved by permitting 20 interconnections per superpeer (40% less time compared to no interconnections). Similar to the number of hops, the response time can be reduced slightly further by allowing an unlimited number of interconnections.

The use of interconnections also shows the desired effect on the response time of *distant area search*, presented in the Figure 53b. The

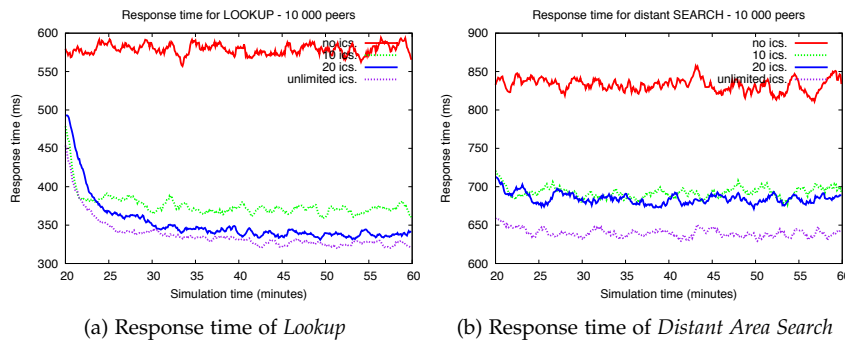


Figure 53: Effects of number of interconnection on response time for queries

average response time to *search* queries, using 10 interconnections, is 17% faster (690 ms on average, compared to 830 ms not using interconnections). Increasing the number of permitted interconnections to 20 does not decrease the response time significantly (an improvement of another 2% can be achieved). Without limiting the number of interconnections in the superpeer tree, average response times of under 650 ms can be reached for *distant area search* in the 10 000 peer scenario which equals an improvement of 23%. The use of interconnections between superpeers in the tree has a strong influence on the performance of the operations in Globase.KOM. Both *lookup* and *distant area search* benefit from interconnections. The number of hops and the average response time are reduced drastically by enabling interconnections (improvements range up to 50% for *lookup* operations).

Figure 54a shows the impact of the use of interconnections in the superpeer tree on relative delay penalty. Without making use of interconnections, relative delay penalty is between 5.0 and 5.3 for the entire time of the simulation. Similar to the response time of the *lookup* operation, the relative delay penalty for the queries starts to fall, when the first user operations are being performed in the scenario. When allowing 10 interconnections per superpeer, the average relative delay penalty drops by 33% to just under 3.5. With the number of interconnections set to 20, a value of 3 can be achieved for relative delay penalty (41% less than without interconnections). A maximum improvement of 45% can be reached when permitting an unlimited number of interconnections.

The effects of interconnections on the load distribution within the superpeer tree can be shown by observing the load balance ratio (shown in Figure 54b). The average load balance ratio when using interconnections is between 2.3 and 2.4 throughout the simulation. Ten interconnections reduce the average LBR to 2.05. Since, in most cases, the most loaded peer can be found in the upper levels of the superpeer tree, it makes sense that using interconnection in the routing process improves the load distribution within the tree. Not having to travel all the way up the tree reduces the load of the root superpeer and the root children. An average load balance ratio of 1.9 can be achieved when permitting the use of 20 interconnections, which corresponds with an improvement

Insignificant performance improvement when having more than 20 interconnections

More than 20 interconnections can negatively influence load balance ratio

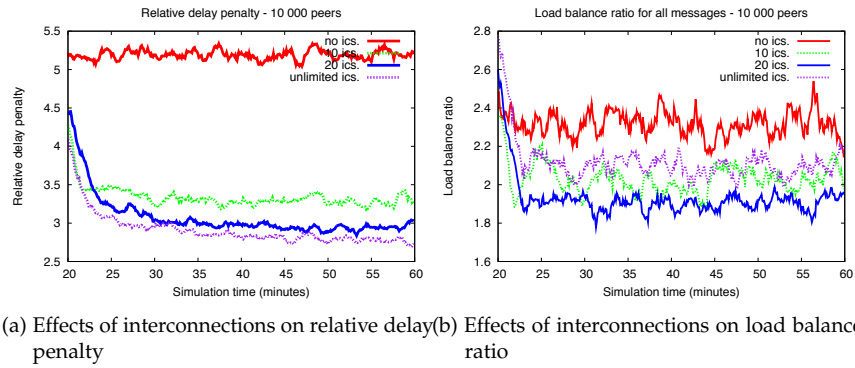


Figure 54: Effects of number of interconnection on costs

of over 20%. Since contacts in the overlay network need to be kept up-to-date, increasing the number of interconnections not only improves the efficiency of the routing process, but also increases the overhead created by maintenance messages. The load balance ratio rises to 2,1 for Globase.KOM, when an unlimited number of interconnections is permitted; exceeding the values for 10 and 20 interconnections. This trade-off has to be considered when deciding about the interconnection parameter S_1 .

Summary

Experiments showed that 20 additional overlay contacts per peer are needed to achieve good improvements in response time for both *lookup* (up to 40% faster response times) and *distant area search* (response times dropping by 20%). In a comparison with Kademlia which has 3131 contacts, this number of additional connections is fairly small. The relative delay penalty is also effected positively by adding interconnections to the superpeer tree. This ratio of the time that a message takes through the overlay compared to the time it takes through the underlying network can be decreased from over 5 without interconnections to about 3 if 20 interconnections are established per peer. Excessive use of interconnections is not advised, as it negatively effects the load balance ratio. Even though interconnections can positively influence the load distribution among the superpeers, as they decrease the load for peers in the upper part of the tree, more overlay contacts create more maintenance overhead. The best results for load balance ratio in this evaluation were achieved with 20 interconnections (load balance ratio improves by 20%). The use of more interconnections results in an average LBR which exceeds the measured values for 10 and 20 interconnections.

6.1.3 General Overlay Parameters

Maintenance, replication, caching related parameters, and operation timeouts are some of the parameter types common to all peer-to-peer overlays. When choosing these parameters, one needs to take into account underlay network condition and user behavior.

The maintenance parameters in our solution are: frequency of periodical failure detection mechanisms and number of missing liveness information. The appropriate parameters values will allow for timely failure detection without causing a noticeable amount of additional overhead traffic. In order to find these values, we observed how the stale message ratio changes with varying frequencies of failure detection messages of different connection types (Section 4.3.6) and number of missing liveness information. Setting these parameters greatly depends on the scenarios of usage and online user behavior. We chose, therefore, the following parameters according to the churn rates and length of experiment timeline we used in our simulations. For almost all superpeer connections, failure detection messages are exchanged every 2 minutes, 1 minute between superpeer and its replica superpeer, and 3 missing liveness information before failure is detected.

Operation timeout should avoid unlimited waiting times for the results of the user operation. It should be long enough to cover the cases of the lost messages and delayed responses. The appropriate value for this parameter depends on the average response time of queries and underlay network conditions. We observed retrievability with varying timeout and choose 2 seconds (double the average response time). For the size of cache, we choose a value which is fixed to 10, derived from the implementations of other overlay networks.

Setting maintenance parameters must take into account underlay network condition and user behavior

Our general parameters' values

6.1.4 Summary

The evaluations of quality properties and comparison of our solution to the representatives of the related work are presented in the following sections. All following simulation experiments are run with the settings previously discussed and chosen, summarized in Appendix G. In the cases that different parameters are used in experiments, we state these values.

6.2 EFFICIENCY

In this Section we evaluate the efficiency of our solution. Efficiency is defined as the ratio of performance and introduced costs. We, therefore, first present the performance of the queries (lookup and area search) and then the costs (protocol overhead). The experiments are run with 10 000 peers, and parameter settings as presented in Table 2. In spite of the fact that Chord [SMLN⁺03] and Kademlia [MM02] were designed for lookup rather than for retrievable search, the performance of their *lookup* query will be used here as reference for a comparison (see Figures 55). In Figure 55a we can see that number of

We compared the performance of 'find on a specific location' queries to the lookup queries of Chord and Kademlia

Protocol parameters		Value(s)
Globase.KOM	Load threshold L_1	55, 38
	Load threshold L_2	110
	Number of interconnections S_1	20
Chord	Number of successors	10
Kademlia	Bucket size (k)	10
	b	5
	Parallelism factor α	3
Simulator settings		Value(s)
Number of peers		10 000
Experiment timeline		Efficiency
Churn		Skype-churn model

Table 2: Parameter settings for the evaluation of efficiency

hops in Globase.KOM is 18% better in the case of parameters $L_1 = 55$ and $L_2 = 110$. Chord needs on average 22.8% more hops per lookup query than Globase.KOM with $L_1 = 55$ and $L_2 = 110$ while Kademlia performs 21% better due to parallel lookup queries and big contact lists. This also reflects the response time (Figure 55b), where Globase.KOM needs 38.4% longer to respond to a lookup than Kademlia. However, the lookup performance difference between Chord and Globase.KOM is even bigger with regard to response time – Globase.KOM performs 53.5% better than Chord. The reason is better underlay-awareness of the Globase.KOM overlay, which significantly reduces RDP (Figure 55c). That is also the reason why both configurations of Globase.KOM have almost the same duration of lookup operation.

As a comparison, a Globase.KOM superpeer has in maximum $L_2(\text{peers}) + 1(\text{root}) + 1(\text{parent}) + \max(5(\text{childrensuperpeers}) + 20(\text{interconnections})) = 147$ contacts, a Chord peer has in average $160 + 10 + 1 = 171$, and Kademlia $155 \times 20 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 3131$ contacts. Even though Globase.KOM has the smallest routing table, it has better (Chord) or just around 20% poorer performance metric values. That means that *lookup* in our solution is more efficient than both Chord and Kademlia as it achieves similar or even better performance for smaller costs in terms of size of routing table.

Performance of *area search* was shown previously on Figures 53b and 53a. For 10 000 peers, for lookup query 350 ms is needed in average, while response time for *distant area search* is 680 ms in average. The number of hops for *lookup* and *distant area search* are presented on Figures 52a and 52b and their average values are 3.5 and 8 respectively. In Section 6.7 we will see the comparison of these values to the related work. Relative delay penalty for *distant area search* (see Figure 54a) is not significantly changed from the case of *lookup* and is around 3 on average. Figure 55d shows that for *distant area search* with 100 peers, in the worst case, only 0.3% of the results are not delivered. This

Comparable hop
count and response
time, but
significantly better
RDP

Performance of area
searches

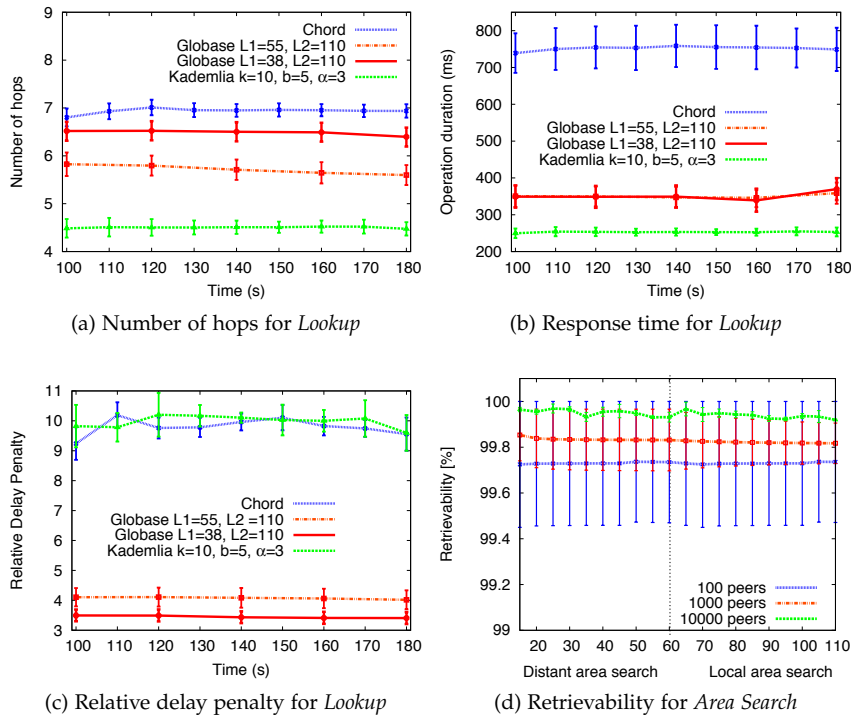


Figure 55: Query performance of Globase.KOM, Chord, and Kademia with 10 000 peers

percentage decreases to 0.1% for 1 000 peers and 0.05% for 10 000 peers. As we can see, there is no difference in retrievability between local and distant area search.

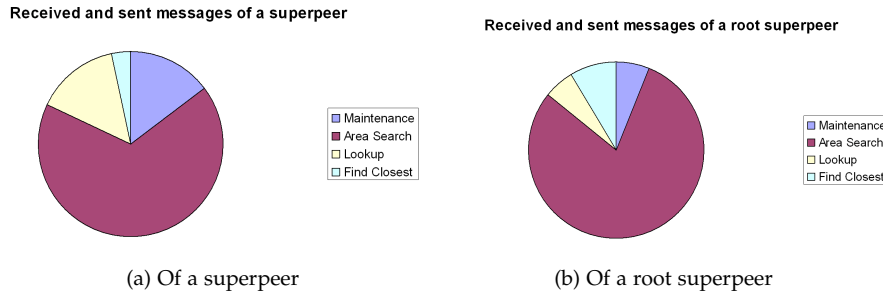


Figure 56: Distribution of incoming and outgoing traffic

In Figures 56 we present the share of overall incoming and outgoing traffic for a superpeer and a root superpeer. Although more than 80% of received messages are maintenance, their size is very small and, therefore, their share in the traffic insignificant (around 20%, maximum 500 Bps for root superpeer, with 10 000 peers in network). Clearly, the share of query messages and their size depends on the scenario and the size of the areas searched for, which influence the size of the results.

Share of maintenance messages large, but uses an insignificant amount of bandwidth

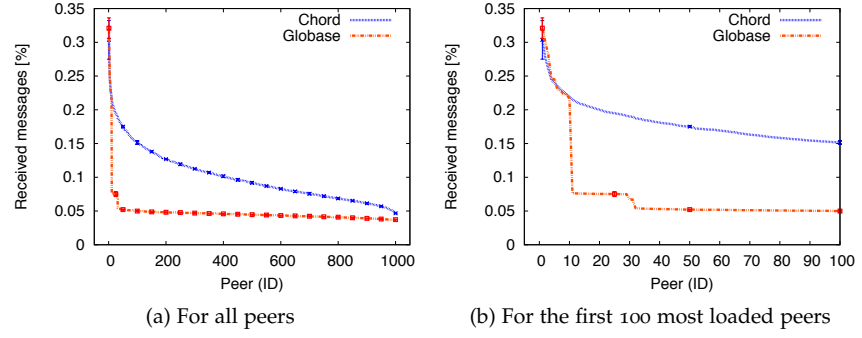


Figure 57: Percentage of received messages

A review of efficiency evaluations

More contacts are included in the results of the area search, the size of the search result messages and their share in the overall traffic is bigger.

Summarized, for evaluations of efficiency, we evaluated Globase.KOM in comparison to Chord and Kademlia. We investigated the performance and costs of lookup, for which Globase.KOM was initially not designed, but is provided by all three overlays. Results show, that Globase.KOM has the least contacts, thus the least costs of the three overlays, but delivers better (in case of Chord) and only slightly worse (in case of Kademlia) lookup performance. The contact-utilization efficiency of Globase.KOM is thus better than that of Chord and Kademlia. Discussing the cost utilization in the overlay, leads to the question: how is the load distributed. We address this question in the next section. Retrievability of distant area search is 99.7% in the worst case and it takes 680 ms on average and 8 hops. Although more than 80% of received messages are maintenance, their size is very small and, therefore, their share in the traffic insignificant (around 20%, 500Bps in the worst case).

6.3 FAIRNESS

Fairness presents the distribution of traffic load (received/sent messages) on the individual peers. The fair load of a peer should be proportional to its individual capacity.

The most loaded peer is 1.9 times more loaded than the median loaded peer

As we can see in the previous evaluations (see Section 54b, Figure 54b), the load balance ratio for the appropriate settings for size of interconnections (20) is below 1.9. That means that the most loaded peer (root superpeer) receives/sends 1.9 times more messages than the median loaded peer. This ratio is even smaller when the peers and superpeers are considered separately.

We observed the percentage of received messages in Chord and in Globase from the experiments for efficiency (however, with 1 000 peers). Figure 57a shows the number of received messages per peer, sorted from the most to the least loaded peer. We can see the load distribution of Globase.KOM and Chord under the identical simulation conditions.

The average load of the peers in Chord is 0.1% and varies between 0.05% and 0.31%. There are no severe differences in load distribution,

though around 40 peers have significantly larger load than other participants. The explanation for this is that in the beginning of the simulation, the Chord ring is built over just a few peers and therefore most of the peers have fingers to those peers. Through stabilization messages, those peers are periodically contacted from all peers which have fingers to them. The average load in Globase.KOM is 0.05% and varies between 0.04% and 0.32%. Figure 57b shows steep load reduction after the first 10 most loaded peers. More exact insight shows that those peers are 10 superpeers, which form the overlay. The larger load of superpeers is due to maintenance messages from child-peers as well as routing messages. The root superpeer has the highest load (0.32%), and its direct child superpeers have around 0.26%. On average, a superpeer receives 0.21% of all produced messages in the overlay. Thus, heterogeneity of the peers is taken into account when selecting superpeers.

The value of the load balance ratio in our solution which is superpeer-based is comparable even with pure peer-to-peer overlays like Chord, which do not take heterogeneity of the peers into account.

Superpeers do not take more load than the most loaded peers in Chord

6.4 SCALABILITY

Scalability is the quantitative adaptability of the overlay to a changing number of participants or services in the overlay, while preserving the performance. With a varying number of peers, the performance should optimally converge to a certain value. We evaluate performance and costs metrics with the different size of the network (see Figures 68, 69). We can see that the response time and relative delay penalty of all types of the queries insignificantly changes with the increasing size of the network. Interconnections and deterministic routing through the tree play a big role in scalability.

Due to greedy routing and long-range contacts, our solution is scalable

6.5 STABILITY

Stability refers to the ability of the system to maintain all functions and services of the system under expected or unexpected conditions and environmental changes. We observed the behavior of the system under frequent leaving (Section 6.5.1) of the peers and intensive queries (Section 6.5.2). The main metric we used is performance variation. Here variation refers to the relative difference of a parameter from its value in the stable state of a system (see Section 5.4). For the stable values we took the average performance and costs values from the experiments on efficiency evaluations. Simulation settings for the experiments on stability evaluations are summarized on Table 3.

Stability evaluated under frequent leaving and intensive queries

6.5.1 Intensive Leaving

We observed the performance and costs variation when 5, 10, 20, 30, and 50% of the peers (regular peers or superpeers) or only superpeers simultaneously leave the network. The increase of query response time

<i>Simulator settings</i>	<i>Value(s)</i>
Number of peers	100, 1 000, 10 000
Experiment timeline	Stability timeline
Churn	Skype-churn model

Table 3: Simulator settings for the evaluation of stability

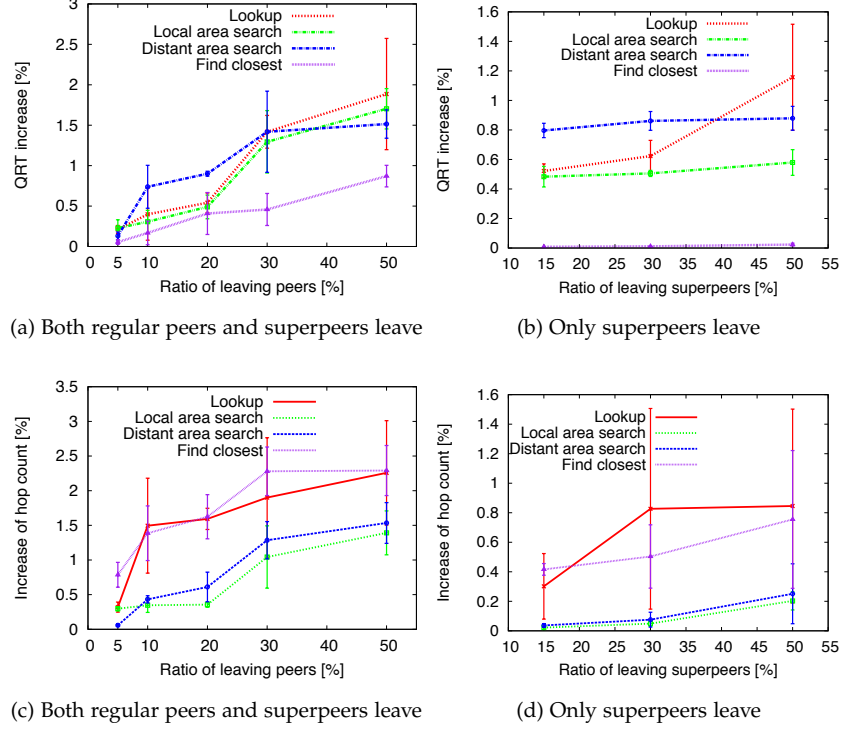


Figure 58: Variation of query response time and hop count when peers/superpeers simultaneously leave the network

*Insignificant
performance
variation when peers
simultaneously leave*

and hop count is shown in Figure 58 when a certain percentage of peers leave and when only superpeers leave.

The query response time increases up to 1.89% when both regular and superpeers leave and up to 1.16% when only superpeers leave. The impact on the *find closest* is minor (maximum 0.009%). As can be expected, the greatest effect of the intensive leaving is on *distant area search* response time as it involves a large number of superpeers in the routing than *lookup* and *find closest*. The increase of hop count in all types of the queries is maximum 2.3% in Figure 58c and 0.8% in Figure 58d.

Retrievability remains 100% in case of *lookup*, *find the closest*, and *local area search*. We present the minor decrease of retrievability of *distant area search* in Figure 59. Retrievability of *distant area search* decreases from almost 0 to 0.0027% when all types of peers are included in leaving and from 0.0003% to 0.0028% when only superpeers leave. That

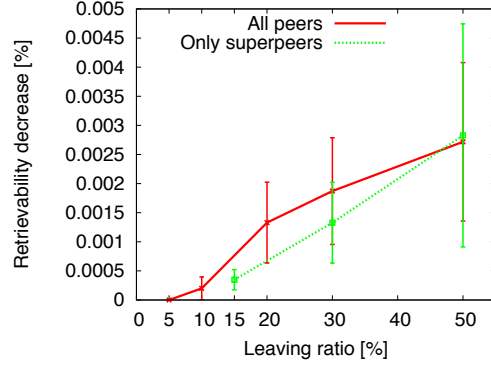


Figure 59: Variation of retrievability when peers/superpeers simultaneously leave the network

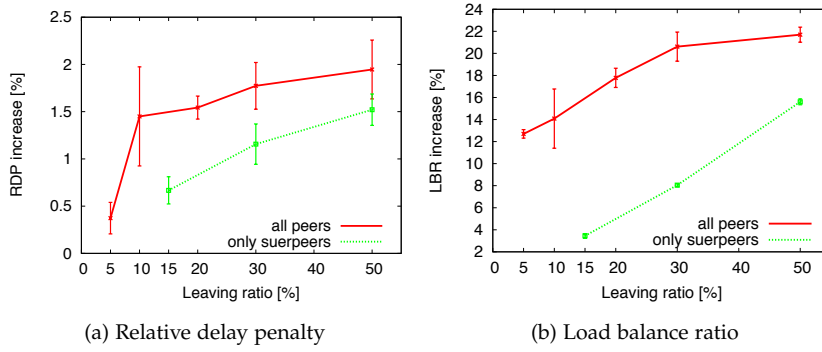


Figure 60: Variation of relative delay penalty and load balance ratio time when peers/superpeers simultaneously leave the network

means that in our experiments with more than 1 000 performed distant area searches, only the result of two did not contain a single existing matching peer contact.

Figures 60 show the increase of relative delay penalty and load balance ratio. Deterioration of relative delay penalty is greater in the case that all types of peers are included in intensive leaving and it is from 0.4% to 1.9%. In the case only superpeers leave, the deterioration is from 0.7% to 1.5%.

Examination of load balance ratio shows the biggest influence of intensive leaving. Figure 60a shows the increase of load balance ratio laying between 12.7% and 21.7% (all peers leaving) and between 3.4% and 15.6% (only superpeers leaving). As this is an insignificant change (maximum load balance ratio was 2.4), we can conclude the stability of load balancing under intensive leaving of peers.

Summarized, in the unlikely case that 50% of peers (or only superpeers) leave the network, the number of hops, query response time, and relative delay penalty will only insignificantly increase (up to around 2%) while the most loaded peer will be around 20% more loaded than in the case of the regular scenario (without simultaneous leaving). Re-

Only load balance is slightly affected, 20% when 50% of peers leave simultaneously

covery time for all of these experiments were constant and its duration was less than a second.

6.5.2 Intensive Queries

Intensive queries = huge amount of queries in a very short time span

In order to simulate possible instability factors, we observed the performance and costs variation during intensive and especially frequent queries. We initiated a huge amount of queries in a very short time span (every 2 ms at least one query is performed, which is 10 times more than in the regular scenario). The frequency of user queries in case of 100, 1 000, and 10 000 peers is 200, 2 000, and 20 000 queries per minute, respectively. Figures 61a and 61b show the increase of number of hops and query response time respectively. It is interesting that the effect of query frequency has a smaller effect on the increasing size of the network. The increase in the hop count per query is 0.44% for 100 peers and 0.02% for 10 000 peers. The more peers are in the network, the better they handle the user requests.

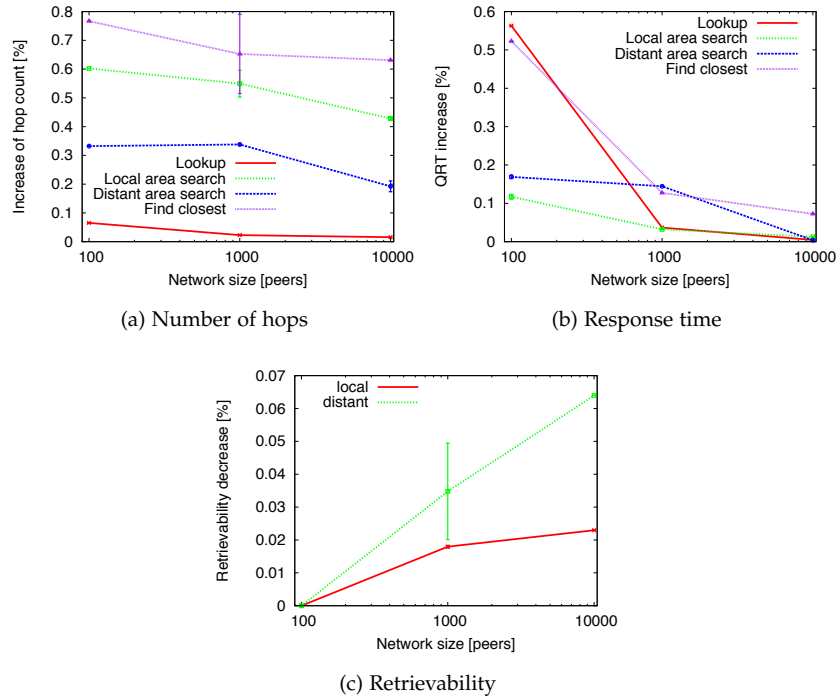


Figure 61: Variation of hop count, response time, and retrievability with more frequent user queries

Insignificant influence of intensive queries on quality

We can see a similar trend in relative delay penalty Figure 62a whose deterioration decreases with the increasing network size. Contrary to this, the variation of load balance ratio (see Figure 62b) increases with the size of the network. It is the result of an increasing frequency of user queries which places the greatest share of traffic on the root superpeer (see Figure 56b).

Retrievability is insignificantly influenced by the increasing frequency of network size (up to 0.06%) and the results are presented in Figure 61c. It is interesting that retrievability is more greatly affected by frequent user queries than intensive leaving of the peers. However, in the experiments with frequent user queries, more queries are performed and there is a greater possibility to have incomplete results.

Summarized, we can see that in the case of frequent user queries our solution is very stable regarding the number of hops, response time, and relative delay penalty. However, the load balance ratio is again the most vulnerable metric. The most loaded peer is up to 9% more loaded than in the case of the regular scenario (used in efficiency evaluations). Recovery time was again shorter than one second and its duration did not vary significantly.

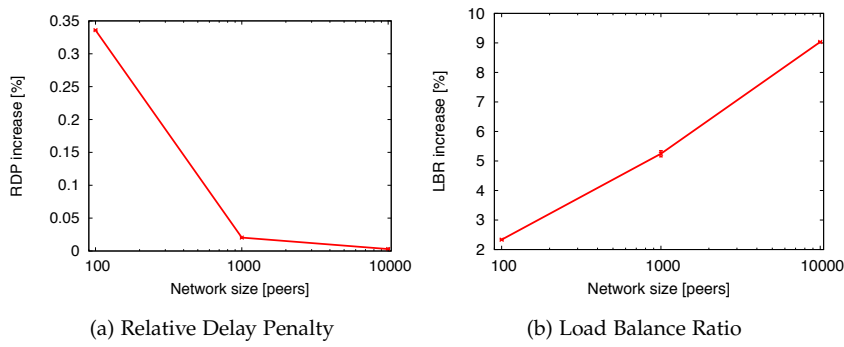


Figure 62: Variation of relative delay penalty, load balance ratio and retrievability with more frequent user queries

6.6 ROBUSTNESS

Robustness describes the variations in performance and costs metrics under critical failures and massive churn rates. We observe the behavior of the system under simultaneous failures of peers (Section 6.6.1), and multiple failures of the superpeers in the same tree branch (Section 6.6.2). The main metric we used was performance variation, because the recovery time was very short and constant for each setting. Like in stability, reference values for variation were taken from efficiency evaluations. Simulation setting for the experiments on stability evaluations are summarized in Table 4.

Robustness evaluated under simultaneous failures of peers on crucial positions in a tree

Simulator settings	Value(s)
Number of peers	100, 1 000, 10 000
Experiment timeline	Robustness timeline
Churn	intensive failures and Skype-churn model

Table 4: Simulator settings for the evaluation of robustness

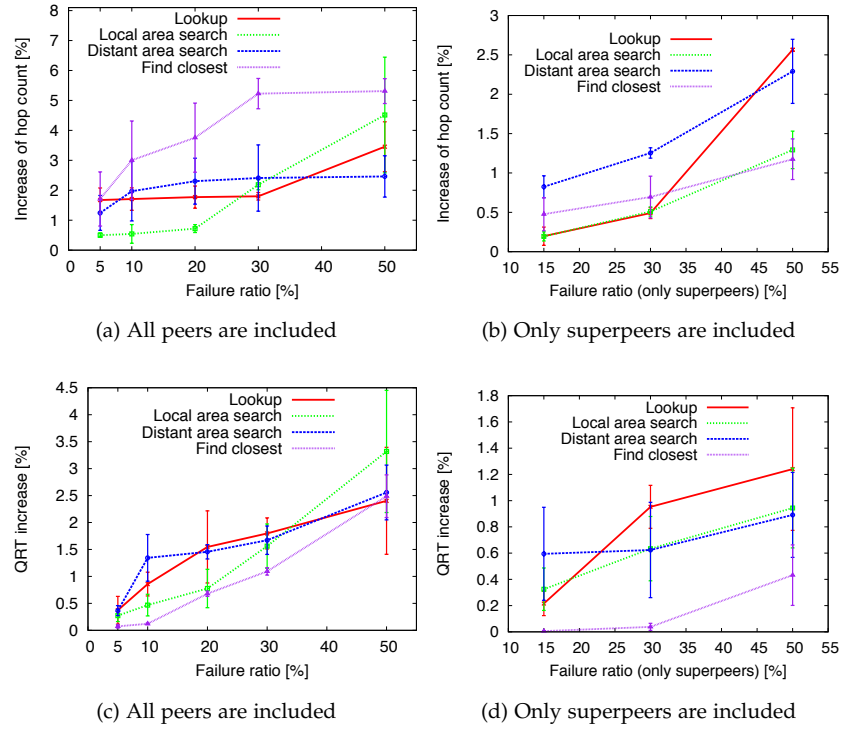


Figure 63: Variation of hop count and response time with intensive peer failures

6.6.1 Intensive Failures

In order to simulate intensive failures, we ran experiments where 5, 10, 20, 30, 50% of any type of peer failed simultaneously and where 15, 30, 50 % of only superpeers failed at the same time. As in stability, we observed the number of hops, response time of the queries, retrievability, relative delay penalty, and load balance ratio.

Figures 63a and 63b show the variation of hop count when a certain percentage of peers or only superpeers simultaneously fail in the case of 1 000 peers. The number of hops increases up to 5%. It is interesting that find the closest query is the most, and distant area search the least, vulnerable to simultaneous failure of peers. The reason is that the resolving of find the closest query relies heavily on the parent-child connections while distant area search uses interconnections. On the other hand, when only superpeers are included in the intensive failures, hop count of distant area search and lookup is the most affected, as their resolving involves longer paths than in the case of local area search and find the closest node (see efficiency evaluations 6.2)

The variation in response time in the case of intensive failures is presented in Figures 63c and 63d. Here we can see the strong effects of underlay-awareness (visible through low relative delay penalty values), as the effects of intensive failures on response time are around half that on hop count. Variation lies between 0.27% and 2.69% (for all peers),

The hop count insignificantly affected even when 50% of peers fail

Variation of response time when certain percentage of peers simultaneously fail

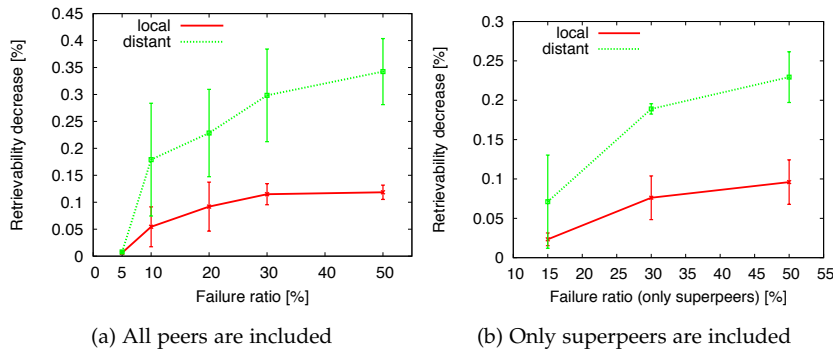


Figure 64: Variation of retrievability with intensive peer failures

and even less for the case only superpeers are included in the intensive failures (between 0.28% and 0.88%).

In Figures 64a and 64b, we can see that simultaneous failures of a large number of peers have again an insignificant effect on retrievability. It is up to 0.36% worse than in the stable scenario, with the normal churn rates.

Figures 65 show the variation of relative delay penalty and load balance ratio, which is again bigger than in stability evaluations. Maximum deterioration of relative delay penalty was 7.58%, while the variation in stability evaluations was only 1.95%. As superpeers might be replaced by a parent superpeers if replica superpeer is not available, the geographical distance between peers and their responsible superpeer gets bigger. That influence on the additional delay that overlay communication brings and decreases underlay-awareness of the overlay.

Remarkable is the variation of load balance ratio in the case of intensive failures of peers, which is between 15% and 30% worse than in the stable scenario. The reasons are uneven distribution of the peers per responsible superpeer and an increased number of exchanged failure recovery messages. A much smaller variation of load balance ratio in the case only superpeers are included in the intensive failures (up to 7%) confirms this argumentation.

Summarized, we observe a greater variation in performance and costs metrics in the case of intensive failures than in stability evaluations. Variation of hop count is up to 5% while response time has double smaller variation, due to good underlay-awareness of the overlay. Retrievability varies insignificantly (up to 0.35%). The relative delay penalty increases approximately 7% as the geographical distance between peers and their responsible superpeers increase during failure recovery. However, the most affected metric is load balance ratio, which experiences an increase of up to 30%. The reasons for this are uneven distribution of the peers per responsible superpeer and an increased number of exchanged failure recovery messages. Nevertheless, as the recovery time is very short, the overlay can be considered robust against simultaneous failures of up to 50% of peers.

*Variation of load
balance ratio is up to
30%*

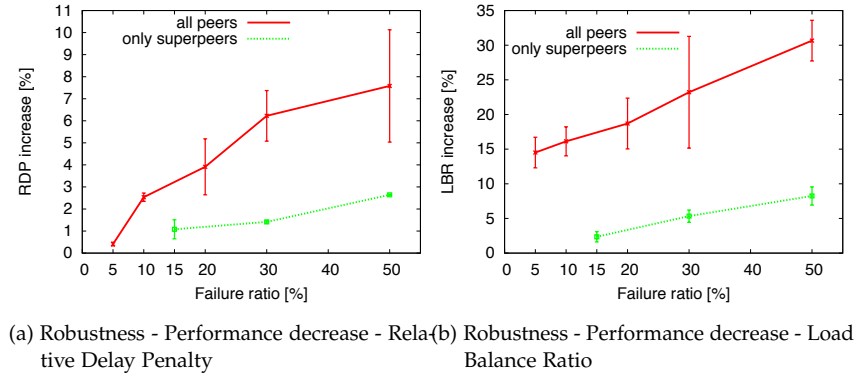


Figure 65: Variation of relative delay penalty and load balance ratio with intensive peer failures

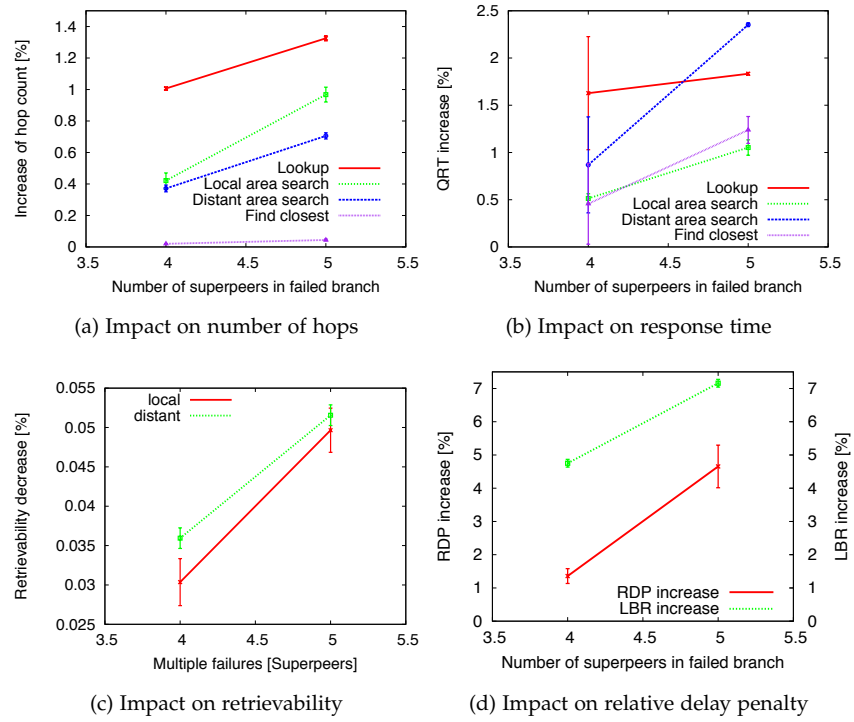


Figure 66: Variation of observed metrics when multiple superpeers from the same tree branch simultaneously fail

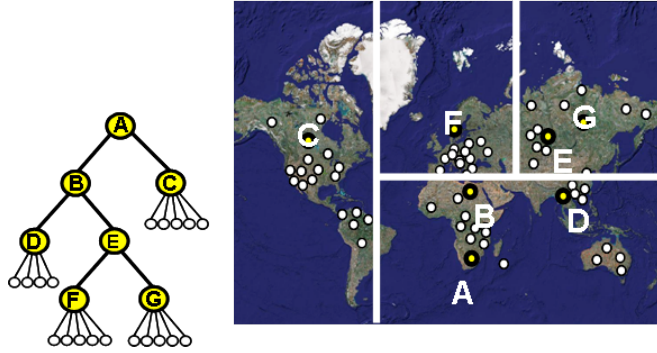


Figure 67: Overlay structure of RectNet (left) and corresponding responsibility zones (right)

6.6.2 Multiple Failures in the Same Tree Branch

Routing and failure recovery in Globase.KOM relies on parent-child superpeer connections. It is, therefore, important to examine the overlay behavior when multiple superpeers from the same branch, connected with direct parent-child connections, simultaneously fail. Here, we show the experiments for 1 000 peers (with $L_1 = 50$ and $L_2 = 100$) where the depth of the tree is usually 5 and there are 15 - 20 superpeers. We therefore evaluate only the simultaneous failures of 4 and 5 superpeers of the same branch (the simultaneous failures of 3 superpeers were included in the previous evaluations). The variations in hop count, response time, retrievability, and relative delay penalty are shown in Figure 66.

The results are comparable to the results with the intensive failures of 30% of superpeers. This shows the obvious robustness of Globase.KOM in the unlikely case when connected superpeers in the same tree branch go offline at the same time.

Simultaneous failure of superpeers in the same branch does not noticeably change performance

6.7 COMPARISON WITH THE RELATED WORK

Previously, we have seen the values of quality metrics in various scenarios which could only give us a clear answer about matching functional requirements ('Does the overlay enable area search, finding a closest peer or a peer in a specific location?'). In order to give statements about non-functional requirements (such as efficiency, scalability, or fairness), we need reference values to the metric values obtained from presented evaluations. That is only possible by comparing the quality of prior related work.

For our comparative evaluations we chose *RectNet* [Heuo5]. It is the overlay most related to Globase.KOM in terms of meeting functional requirements (fully retrievable area search) and design goals. Like our approach, it supports area search, finding the closest peer and peer in a specific location. RectNet is an implementation of **Distributed Space Partitioning Tree** (DSPT). The structure of the overlay and the

We compare our solution with RectNet

<i>Simulator settings</i>	<i>Value(s)</i>
Number of peers	100, 1 000, 10 000
Experiment timeline	Efficiency timeline
Churn	Skype-churn model

Table 5: Simulation settings for the comparative evaluation of Globase.KOM and RectNet

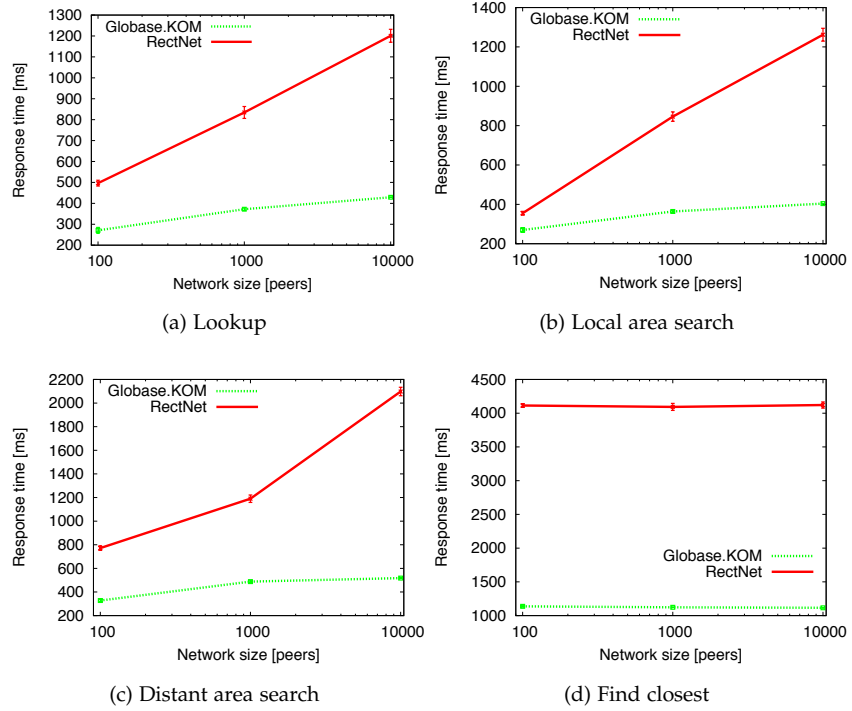


Figure 68: Response times of Globase.KOM and RectNet

corresponding responsibility zones are depicted in Figure 67. It is a hybrid, tree-based overlay, similar to our solution. However, superpeers build binary tree and all results of the queries can only be collected at the leaves of the tree. This can lead to longer routing paths and increased response time in comparison to Globase.KOM. Additionally, contrary to our solution, peer identifier space does not include information about responsibility area nor peer location. This introduces additional communication (in order to check the responsibility of the contacted peer) and does not allow greedy routing. A more detailed description and analysis of this approach is given in Section 2.

We ran experiments with the settings presented in the Table 5. We evaluated four types of user queries: finding a peer in a specific location (*lookup*), finding all peers in the surrounding local area (*local area search*) and in distant areas (*distant area search*), and finding the closest peer to the given location (*find closest*).

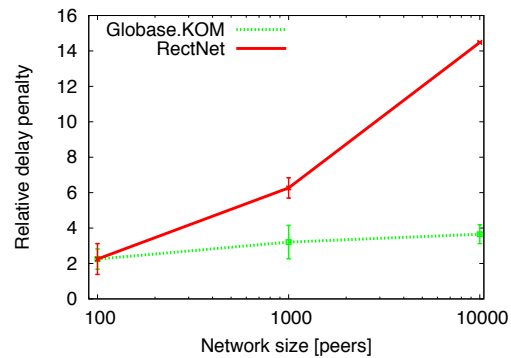


Figure 69: Comparison of Relative Delay Penalty

Figures 68 show the comparison of response times of all four types of queries. We notice the evident difference between two solutions - Globase.KOM performs two to three times faster lookups and local area search, three to four times faster distant area search and four times faster find closest query. The huge impact of peer identifier space in Globase.KOM, management of identifier space, and interconnections is visible here. Additionally, the response time of RectNet increases dramatically with the increase of network size, showing scalability issues. On the other hand, response times of Globase.KOM change logarithmically with the size of the network. The good scalability of our solution is the result of carefully chosen interconnections, routing strategy, and zone forming.

Evident performance difference

Figure 69 shows a comparative evaluation of relative delay penalty during all experiments. We can see a big difference in the case of 1 000 (three times bigger than Globase.KOM) and 10 000 peers (more than 4 times). This is the result of the non greedy routing strategy, caused by underlay-unaware management of identifier space. Query messages must always travel upwards in the tree, unnecessarily involving superpeers that are geographically very distant. Routing strategy does not allow bypassing of the higher levels of the tree. Therefore, the delay introduced by using connections in RectNet overlay are significantly (14 times for 10 000 peers) bigger than using the shortest underlay connections.

Our solution achieves better underlay-awareness

Our evaluations showed comparable values of quality metrics for load balance, robustness, and stability. It is not surprising that RectNet is very robust and stable, as the strict structure of the binary tree result in fast maintenance process. Superpeers in the higher levels of the tree need to maintain the connections to just three other superpeers (parent and two children superpeers) and their replacement in the case of failure is trivial using the appropriate replication mechanisms. More surprising are the good evaluation results of load balance in RectNet. As the query messages need to be routed from leaves to the root and again to leaves in other tree branches, we can expect that the superpeers in the higher levels are performance bottlenecks and experience overload. It seems, however, that this overload caused by

Comparable fairness, robustness, and stability

query messages is neutralized by the fact that superpeers in the higher levels of the tree are not responsible for any peers, but just for three superpeer connections.

Summarized, our comparative evaluations showed evident superiority of Globase.KOM in efficiency and scalability. It performs queries approximately 4 times faster and the additional delay introduced by overlay routing compared to the shortest underlay routes is 4 times smaller than in the case of RectNet. The effect of the choice of identifier space in our solution was evident as it allows efficient, greedy routing. Using interconnections allows bypassing of the superpeers in the higher tree levels and decreases the response time of the queries. The choice of management of identifier space and routing strategy are reflected in better underlay-awareness. In the terms of load balance, robustness and stability, the evaluation results of both solutions are comparable, and their difference, insignificant.

6.8 ANALYSIS OF REQUIREMENTS REALIZATION

After all evaluation results are discussed in previous sections, we must review the requirements for peer-to-peer location-based search stated in the beginning of this thesis, in Section 1.2. This section answers the question of how good these requirements are met, based on the presented evaluation results.

Finally, does our solution fulfill given requirements?

Functional requirements are providing fully retrievable:

- area search,
- finding a peer in a specific location (referred as lookup), and
- finding the peer closest to a given location (referred as find the closest).

This is assessed by retrievability of the queries. Our results showed that in realistic workloads it remains at almost 100% (i.e. precisely 99,7%) and reaches its minimum of 95% in the unlikely case when 50% of the peers simultaneous fail. Therefore, we can state that our solution fulfills functional requirements.

Non-functional requirements:

- The *efficiency* of all query types is significantly better than the efficiency of the related solution RectNet (e.g. queries 4 times faster). The performance of lookup is better than the most referenced overlay for lookup, Chord, and just around 20% worse than in Kademlia. Our solution is, however, more efficient as it achieves similar or even better performance for smaller costs in terms of size of routing table (average size of routing table for Globase.KOM is 147, Chord is 161, and for Kademlia is 3131). Additionally, the underlay-awareness of Globase.KOM proved to be the largest among all of these three reference overlays.
- Although *load-balancing* seems to be an issue due to the tree structure of the overlay, the value of its metric is comparable even to

the pure peer-to-peer overlays that do not assign additional responsibilities to more capable peers. Interconnections and careful selection of parameters for zone forming diminish the expected overload of superpeers in the higher tree levels.

- The *scalability* of our solution is logarithmic and is better than that in the related solution RectNet. This is due to the interconnections, choice of peer identifier space, and greedy routing.
- Our solution proved to be very *stable* under simultaneous leaving of a large portion of the peers and frequent querying. The variation of quality metric is insignificant (up to 2%) except in the case of fairness, where it becomes ca. 20% worse. However, the recovery time is under one second.
- Under simultaneous failures of a large portion of peers, superpeers from the higher levels, or in the same tree branch in our solution prove to be very *robust*. The variation of quality metrics was insignificant (retrievability up to 0.35%, other metrics up to 7%), except again, the load balance ratio, which experiences an increase of 30% when 50% of the peers simultaneously fail. Nevertheless, the recovery time is very short.

The biggest influence on the quality of our solution was the choice of identifier space, its management, and interconnections. A peer identifier contains the information about responsibility of the peer and its location. This allows smart selection of interconnections and efficient greedy routing. Management of identifier space allows good underlay-awareness of the overlay and therefore faster query response time. Interconnections enabled bypassing of the superpeers in higher levels of the tree and therefore allowed equal load distribution among the superpeers. Fast recovery time and small performance variations under extreme churn and critical failures is to be credited to the various maintenance strategies used in combination.

Effects of our design decisions

6.9 SUMMARY

This chapter presented and discussed the evaluation results of our solution, Globase.KOM corresponding to the evaluation goals and methodology from Chapter 5.

We first presented the calibration of the system-wide parameters: the load threshold parameters L_1 and L_2 , number of interconnections, size of cache, timeouts of operations, frequency of periodical failure detection messages, and the number of missing periodical liveness information before a failure is detected.

The evaluation results showed that the value of load threshold L_2 effect the breadth, while the ratio $\frac{L_1}{L_2}$ influence depth of the superpeer tree. It was shown that the shape of the tree has significant impact on fair distribution of the load among the superpeers. A well balanced tree provides good performance and costs values. We chose 0.5 as a

Parameters calibration

A review of effects of L_1 and L_2

good value for ratio $\frac{L_1}{L_2}$ that have been proven to produce good performance results together with proper load distribution among the superpeers. We saw that greater values of L_2 , in relation to the size of the network, result in a better load balance. However, setting this parameter as relative to network size, introduce a additional issues concerning reforming the existing zones and assigning huge numbers of peers to the superpeers. We, therefore, applied two optimizations regarding the formation of the zones of the root superpeer and connections between the superpeers in the second tree level. We limited the number of inner zones of root superpeers and formed zones larger than a hotspot by using maps of possible peer placement (good example is a map of Skype or Internet users). Additionally, direct child superpeers of the root superpeers are connected via the interconnections which significantly decrease the load of the root superpeer. We set L_2 to be 100 and 120 with these two optimizations of the first tree level.

A review of effects of interconnections

Experiments showed that only a fairly small number of 20 interconnections per peer is needed to achieve cost-effective improvement in response time for both *Lookup* (up to 40% faster response times) and *Distant Area Search* (response times dropping by 20%). The relative delay penalty is also effected positively by adding interconnections to the superpeer tree. It was shown that interconnections positively influence the load distribution among the superpeers, enabling bypassing of the superpeers in the higher tree levels. However, having too many interconnections introduce more maintainance overhead. The best results for load balance ratio in this evaluation were achieved with 20 interconnections (load balance ratio improves by 20%).

A review of setting general overlay parameters

Maintenance, replication, caching related parameters, and operation timeouts are some of the parameter types common to all peer-to-peer overlays. The choosing of these parameters needs to take into account underlay network conditions and user behavior. For the simulations we ran, appropriate churn rates, and experiment timeline length, we chose 2 minutes for almost all superpeer connections, 1 minute between a superpeer and its replica superpeer, and 3 missing liveness information before failure is detected. Setting these parameters greatly depends on the scenarios of usage and online user behavior. We observed retrievability with varying timeout and chose 2 seconds (double the average response time). For the size of cache, we chose a value which is fixed to 10, derived from the implementations of other overlay networks.

A review of efficiency evaluations

Further, we examined efficiency, fairness, scalability, stability, and robustness. In spite of the fact that they are design for lookup rather than for search, we used Chord [SMLN⁺03] and Kademlia [MM02] for evaluations of lookup. Chord needs on average 22.8% more hops than Globase.KOM, while Kademlia involves 21% fewer peers in route due to parallel lookup queries and large contact lists. It is similar is with the response time of the queries, Globase.KOM showed evident supremacy in underlay-awareness, having significantly lower relative delay penalty values. Our solution is, however, more efficient as it achieves similar or even better performance for smaller costs in terms of the size of the routing table (average size of a routing table for Globase.KOM is

147, Chord is 161, and Kademlia is 3131). Retrieval of distant area search is 99.7% in the worst case and it takes 680 ms on average, and 8 hops. Although more than 80% of received messages are maintenance, their size is very small and therefore their share in the traffic, is minor (around 20%, 500Bps in the worst case).

Our examinations of fairness showed that the most loaded peer (root superpeer) receives/sends 1.9 times more messages than the median loaded peer. This ratio is even smaller when the peers and superpeers are considered separately, and the ratio is comparable to the flat, pure peer-to-peer overlays.

A review of fairness evaluations

Stability evaluations showed that in the unlikely case that 50% of peers (or only superpeers) leave the network, the number of hops, query response time, and relative delay penalty will only insignificantly increase (up to around 2%) while the most loaded peer will be around 20% more loaded than in the case of regular scenario (no simultaneous leaving). Recovery time for all of these experiments was constant and its duration was less than one second. We saw that in the case of frequent user queries, our solution is very stable regarding all metrics.

A review of stability evaluations

In examinations of robustness, we observe bigger variation of performance and costs metrics in the case of intensive failures than in stability evaluations. Variation of hop count is up to 5% and response time increases 2.5%, due to good underlay-awareness of the overlay. Retrieval varies insignificantly (up to 0.35%). Relative delay penalty increases about 7% as the geographical distance between peers and their responsible superpeers increase during failure recovery. The most affected metric is, however, load balance ratio, which experiences an increase of up to 30%. The reasons are uneven distribution of the peers per responsible superpeer and increased number of exchanged failure recovery messages. Nevertheless, as the recovery time is very short, the overlay can be considered robust against simultaneous failures of up to 50% of peers. Globase.KOM proved to be robust in the unlikely case where connected superpeers in the same tree branch go simultaneously offline.

A review of robustness evaluations

Finally, we ran the comparative evaluations between our solution and RectNet, as the most related overlay in terms of meeting functional requirements and design goals. Evaluation results showed an evident superiority of Globase.KOM in efficiency and scalability. It performs queries at approximately 4 times faster and the additional delay introduced by overlay routing compared to the shortest underlay routes is 4 times smaller than in the case of RectNet. The effect of the choice of identifier space in our solution is evident as it allows efficient, greedy routing. Using interconnections allows bypassing of the superpeers in the higher tree levels and decrease the response time of the queries. The choice of management of identifier space and routing strategy reflect a better underlay-awareness. In the terms of load balance, robustness and stability, the evaluation results of both solutions are comparable and their difference insignificant.

A review of comparative evaluations

PROOF OF CONCEPT

After we presented the design and evaluation of our solution for peer-to-peer location based search, we must additionally prove the viability, protocol completeness and identify practical operational and deployment issues by providing a proof-of concept. This chapter introduces a reference prototype implementation of the Globase.KOM overlay, described in Chapter 4. After a short description of the prototype user interface in Section 7.1, we describe the prototype implementation design in Section 7.2.

Moving beyond simulation: a prototypical implementation

7.1 PROTOTYPE USER INTERFACE

We introduce our prototype by presenting its user interface. A user can perform *area search*, *find closest*, and *find exact* using a given interface in the *search* panel (see Figure 70).

GUI for performing location-based searches

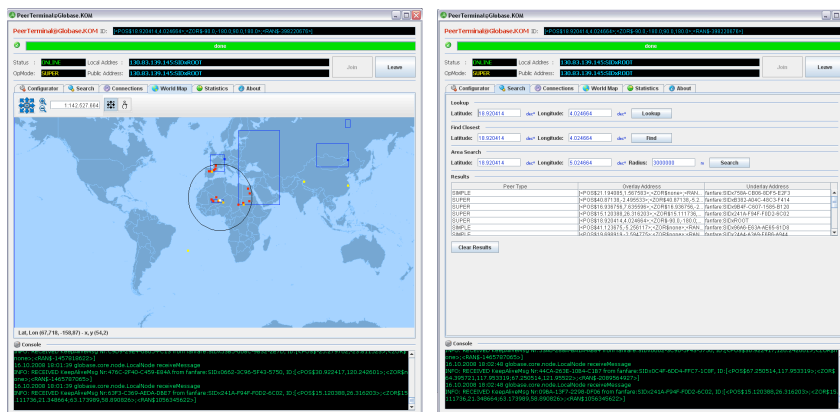


Figure 70: World map and search panel of our prototype user interface

A query is specified by latitude and longitude of a location, and the radius of a searched area. In the same panel (see Figure 70b), the overlay and underlay address and the types of peers from the search results are listed. The position of the found peers, searched area, neighbors and their zones are drawn on a world map in a *world map* panel. Overlay and underlay addresses of neighbors and the metadata describing their services are displayed in the *connections* panel, which is read-only. All system-wide parameters (e.g. load thresholds, number of interconnections, timeouts) can be set by manually entering values in the configuration interface or loading the appropriate xml file in the *configuration* panel. A user can see the amount of messages received recently and per second on average, separated into the user and system messages *statistic* panel (see Figures 71a and 71b). There are two

Four tab panels of GUI

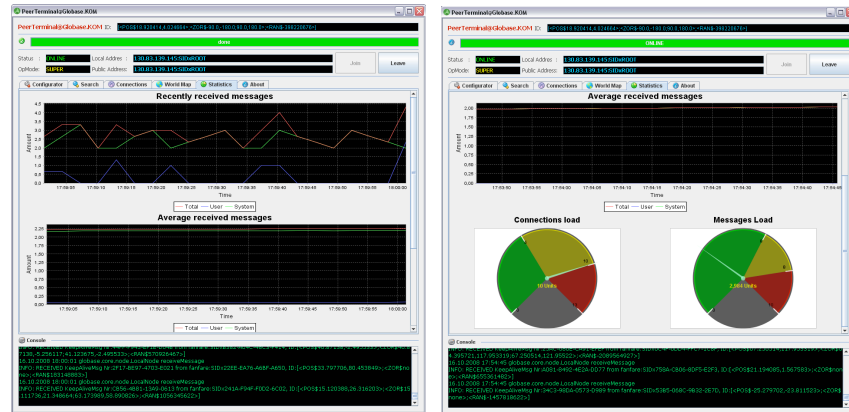


Figure 71: Statistics panel of our prototype user interface

barometers indicating the current load of the peer (based on the two load criteria considered in the prototype) – amount of peers in the zone of responsibility and amount of received messages per second. Diverse libraries from the OpenMap2 project [Ope] for world map panel to the JFreeChart 3 [JFr] libraries for statistic panel were used.

7.2 DESIGN

This section provides software architecture details, starting with the chosen programming language and network communication framework. Due to its platform neutral characteristics, we chose the Java programming language for development. We utilized Java RMI (Remote Method Invocation) [RMI] to realize network communication, which is built on a top of the TCP/IP protocol. We implemented the RMI calls: non-blocking and asynchronous, using threads. This allows independent simultaneous accepting, processing, and issuing of requests. The drawback of calling a remote method using RMI is that it involves unnecessary additional messages. Once a RMI connection is established, however, a remote called method cannot get "lost", like a simple TCP message can. This eases implementation and debugging of our prototype. This network communication mechanism could be exchanged without invasive code change due to our modular design.

7.2.1 Architecture

The architecture of the Globase.KOM prototype consists of three main components: *core*, *extensions*, and *gui*. Figure 72 shows the architecture of the prototype implementation. The core component contains all necessary communication implementations and is organized in seven further components.

The *globase.core* package contains the full implementation (Sections 7.2.3 and 7.2.6) of the Globase.KOM overlay. All messages spec-

*We use Java and RMI
build on TCP/IP*

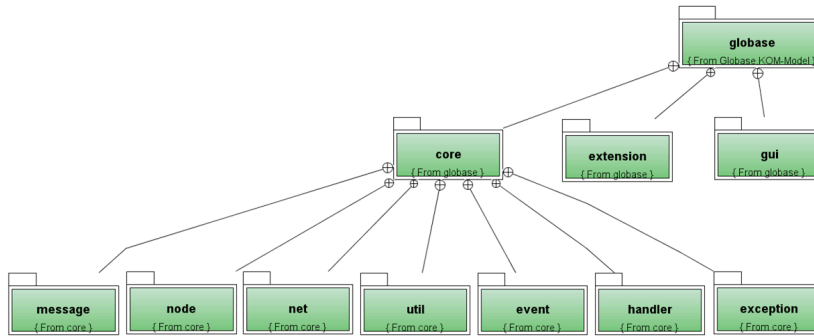


Figure 72: Package organization of the prototype

ified in the protocol are implemented in the `globase.core.message` package (Section 7.2.4). The `globase.core.net` package contains the interfaces (and all their implementations) used by the overlay to communicate with the underlay network (Section 7.2.2). Many important utilities and structures used by the overlay, such as GIS data processing, distance calculation and serial number generation, are offered by the `globasebase.core.util` package. Other important packages are the `globase.core.event` and the `globase.core.handler` package. The first one contains all the events that the prototype generates or may generate at runtime (Section 7.2.7), and the latter holds all message handlers (Section 7.2.4). User interface is implemented in the `gui` package.

To achieve modularity, we defined three functional layers, whose dependencies are kept to a minimum. These are illustrated in Figure 73. This way, it is possible to isolate network logic, overlay logic and application from one another. The layers from the top to the bottom are:

The basic architecture of a prototype

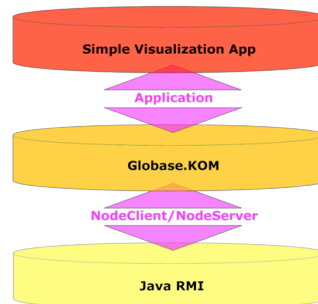


Figure 73: The three main functional layers and interfaces between them

- an *application* which is, in our case, a simple application with a visual interface, allowing a user to perform the basic location-based queries,
- the *overlay* layer, the implementation of the overlay communication protocol, algorithms and mechanisms (in our case `Globase.KOM`), and

- the *underlay* layer, which encloses the logic used to transport the messages through the physical network. Sample underlay implementations could use sockets, http requests/responses, remote method invocations, or some other technique. As already mentioned, our underlay implementation is based on RMI and uses TCP/IP for data transfer.

In order to achieve maximal isolation between the three layers, strict interfaces for the interaction between these components were defined. Both interfaces will be described in the following sections (7.2.2 and 7.2.3).

7.2.2 Underlay-Overlay Interface and Network Layer

For the interaction between the *underlay* and the *overlay*, two interfaces were specified (see Figure 73). It is assumed that, regardless of the overlay implementation, a peer will always be a client and a server at the same time. There is, therefore an interface for the client aspect of a peer called `NodeClient`, and an interface for the server aspect called `NodeServer`. The `NodeClient` interface specifies methods for es-

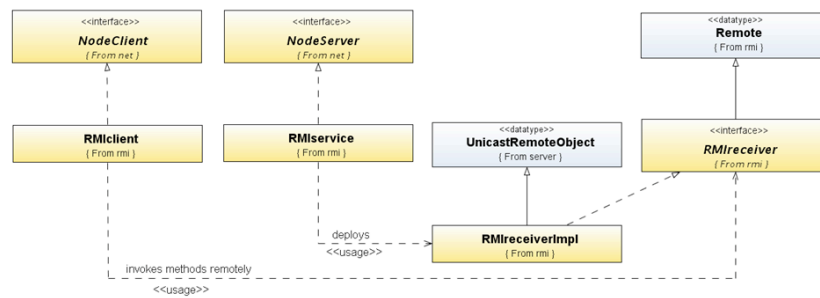


Figure 74: Architecture of the network layer in the Globase.KOM prototype

establishing the connection to a remote party and sending messages (e.g. `send(Message, Address)`). The `NodeServer` reacts to incoming connection requests and sends messages. As already mentioned, the current implementation of these two interfaces is Java RMI-based. Therefore establishing a connection to other peers in the `NodeClient` implementation (`RMINodeClient`) is done by connecting to their `RMIRegistry` and invoking remote calls through the provided Java RMI stubs. The `NodeServer` implementation (`RMIService`) deploys the services of a peer by starting the `RMIRegistry`. The architecture of the network layer is presented in the Figure 74).

7.2.3 Interface between Overlay and Application Layer

In this section, the interface used for interaction between the *overlay* and the *application* layer is described (see Figure 75). It is an abstract class `globase.core.Application`. By extending this abstract class, an application can use the services offered by the overlay (Globase.KOM).

Underlay-overlay
interface

It offers all functionalities of the overlay and leaves the application specific part abstract.

Application and overlay layer separated

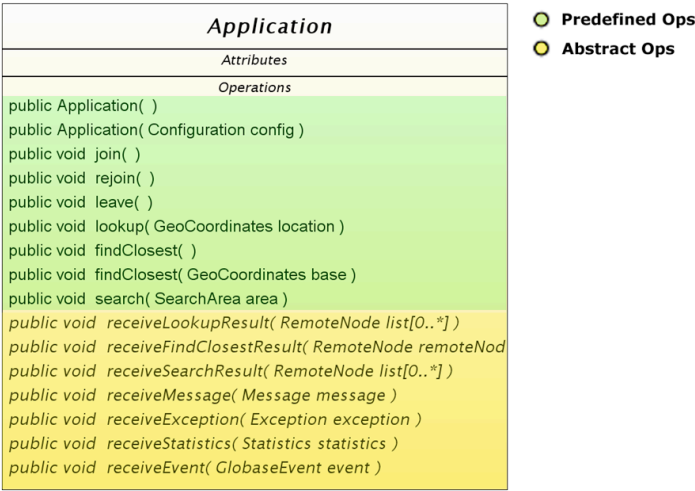


Figure 75: Interface between overlay and application - globase.core.Application

7.2.4 Messages

The Globase.KOM overlay protocol uses a number of different message types. All of these types extend the general type called *Message*, which contains basic information such as receiver ID, initiator ID, timestamp, etc. There are three main categories of the messages described by overlay protocol

Messages: notifications, requests, responses

- *Notifications* - these are messages that are used to inform the receiving party of a specific event and do not require a reply,
- *Requests* - these messages are used to request some information from the receiving party and a reply is expected,
- *Responses* - are messages sent back as a reply to a previously received request.

The full list of the messages are given in Appendix H.

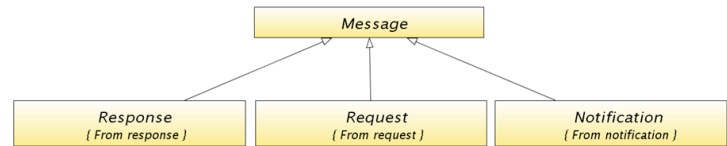


Figure 76: The three generic message types in the Globase.KOM prototype

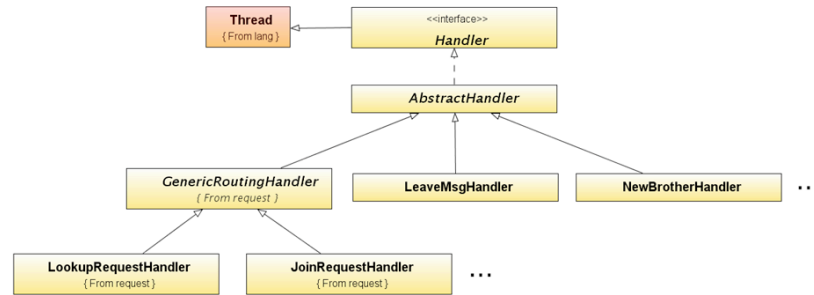


Figure 77: Handlers in the Globase.KOM prototype

Handlers

For every message type in the prototype, a dedicated *Handler* was specified. They are implemented as threads to enable handling of concurrent requests (see Figure 77). The handlers are typically created and started at message arrival.

7.2.5 Message Processing

A *Message* is sent by invoking a remote method over RMI on the receiving peer with the message as a method argument. This method does not block the main thread as it is invoked from a separate thread. To keep the threads lifetime as short as possible, the remote party invokes the return statement right away and then starts a handler thread for processing the message. If the message requires a reply, when the results of the processing are ready, they are sent back to the requesting party in the same manner.

A sample flow diagram that illustrates the control flow, creation, and timeout of the handlers, is given in Figure 78. Here the Actor, which typically interacts with an *Application*, invokes a lookup request. The *Application* packs the actor's request into a *LookupRequest* message and forwards it to the *LocalNode*. The message is then sent through the *NodeClient* interface, whose RMI-based implementation invokes a remote method on the next hop's *NodeServer* interface. When the message arrives at the intended destination, the *NodeServer* interface of the receiver sends it locally to the *LocalNode* object, representing the destination. In order to finish the remote method as soon as possible, the appropriate *Handler* is started in a separate thread. In our case the *LookupRequestHandler* implementation is invoked, which is marked red in the figure. While the handler is busy processing the message, all pending local and remote methods return and thus unblock the threads invoking them. After the handler at the remote party (blue in the figure) has finished processing, it sends back the result in the same manner. The peer initiator handles the message and forwards the result to the *Application* and thus to the Actor.

Control flow at
message arrival

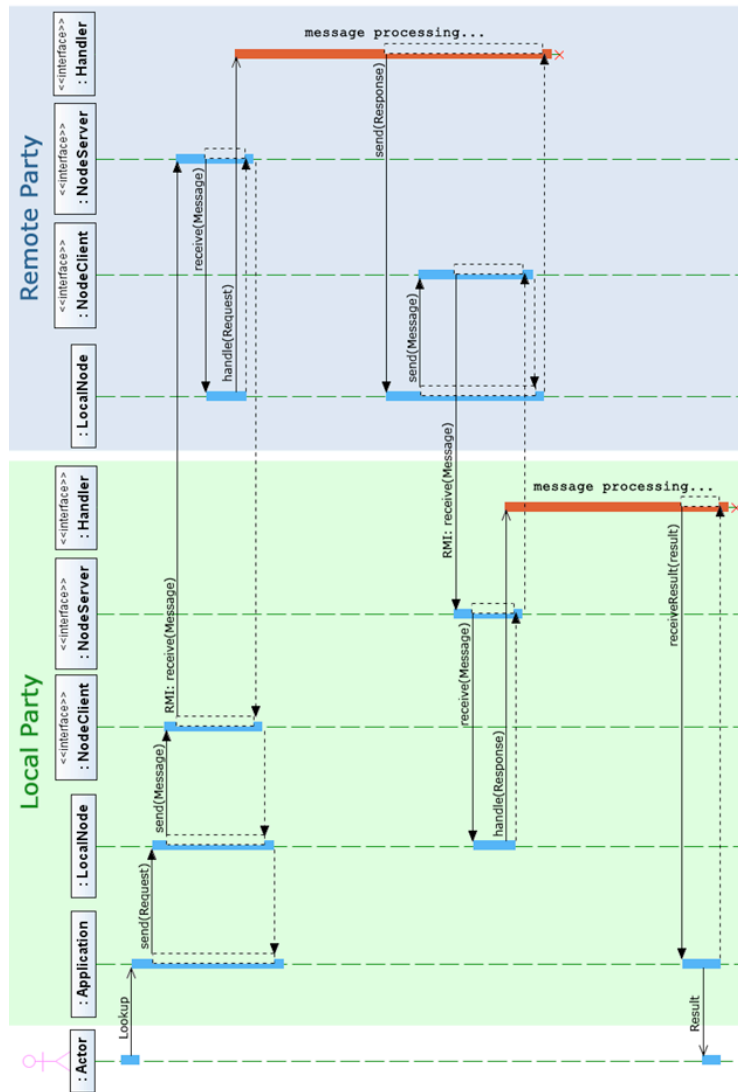


Figure 78: Sample control flow at message submission/arrival

7.2.6 Routing Tables

The routing table of a peer includes contact and metadata information about its neighbors (according to the overlay description in Section 4.3.4) and is organized and stored as shown in Figure 79. The class `NodeID`, represents the IDs of the single peers. It defines a structure to store and retrieve the single elements of an ID, such as position, zone of responsibility and a random part (see Section 4.3.1). The class `PhysicalAddress` represents the underlying network address of a peer, and the class `Metadata` stores metadata description of the offered resources of a peer.

The combination of this information is stored by the class `RemoteNode` and is sufficient for representing and contacting a peer. All neighboring

*Implementation of
routing tables*

peers are stored and maintained by instances of this class (i.e. all interconnections, children, the parent and the root peer) and are organized in different tables. The class `LocalNode` represents the local running instance of Globase.KOM and is stored as a reference for other peers.

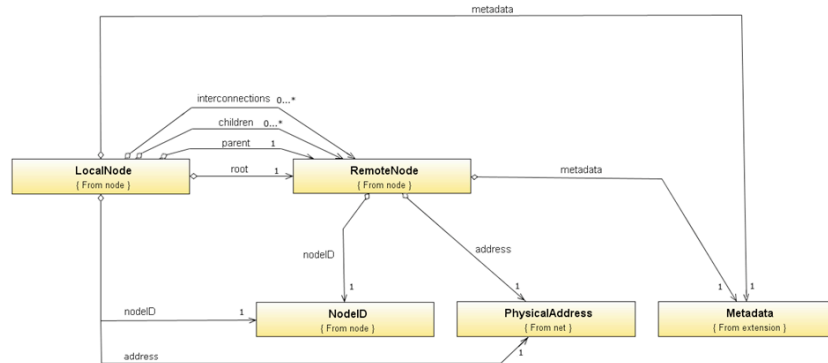


Figure 79: Collaboration between the main elements involved in the representation of interconnections

7.2.7 Events

Events in prototype

Important status changes in the network are announced by the events which are received by the application (see Figure 75);

- *PeerRemovedEvent* - the running instance of the prototype removes a peer of its tables.
- *PeerAddedEvent* - the prototype instance adds a peer to its tables.
- *OpModeChangedEvent* - the operation mode of the prototype instance changes: SUPER (indicates that the running instance is a superpeer) or SIMPLE (if it operates like an ordinary peer).
- *StatusChangedEvent* - the status of the prototype changes: ONLINE, OFFLINE, or DISCONNECTED (indicating that the connection to the peers parent is lost).
- *LookupStartedEvent* - a lookup query starts.
- *FindClosestStartedEvent* - a find-closest query starts.
- *AreaSearchStartedEvent* - a area-search query starts.

The events are shown in Figure 80.

7.2.8 Statistics

Similar to the events, basic statistical information is pushed periodically to the *Application*. The collected statistics are the average amount of all received messages, classified into user and system messages, and

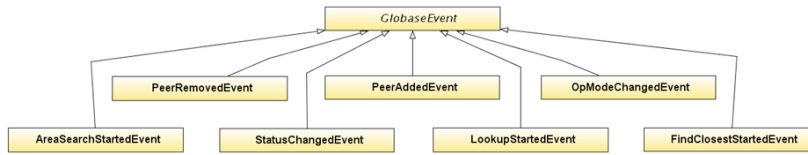


Figure 80: Events in the Globase.KOM prototype

the load level of the instance. The implementation is extendable to provide further statistical measurements, such as the up- and download bandwidth usage, packet size, transmission times, etc.

7.2.9 Daemons

For monitoring the network status and executing the maintenance operations, we implemented daemons – separate thread (shown in Figure 81) that run parallel to the prototype’s main thread. While the main thread is busy serving the overlay, the application, and interactions with the user, the daemons monitor the state of the prototype, ensure proper workflow, and issue alerts or perform specific actions, when the current state or workflow is disturbed. The non-blocking nature of the prototype additionally requires the use of daemon threads, which monitor for lost or undelivered responses, since the thread issuing the request has no knowledge of when, and if a response is going to arrive. Currently, there are three main daemon threads running parallel to each instance of the prototype:

Daemons – separate threads to the prototype main thread

- a *ResponseTimeoutDaemon* monitors and queues all submitted requests and the reception of their corresponding responses. If a reply on an issued request is not received after a timeout, the daemon informs the initiating instance.
- a *KeepAliveDaemon*, that sends the periodical *keep-alive* messages (described in Section 4.3.6) and receives these messages from other peers. Dead peers are detected if no message is received within a configurable time interval.
- a *LoadMonitorDaemon*, used to monitor and control the local load of the peer, based on the pre-configured parameters L1 and L2 (discussed in Sections 4.5 and 6.1.1). The algorithms for load balancing are invoked if necessary.

7.2.10 Geolocation

We implemented several geolocation components, which are in charge of retrieving a geolocation:

How do the peers know their location?

- *RandomLocationGenerator* - this is a simple generator that generates valid random coordinates, regardless of where on Earth.

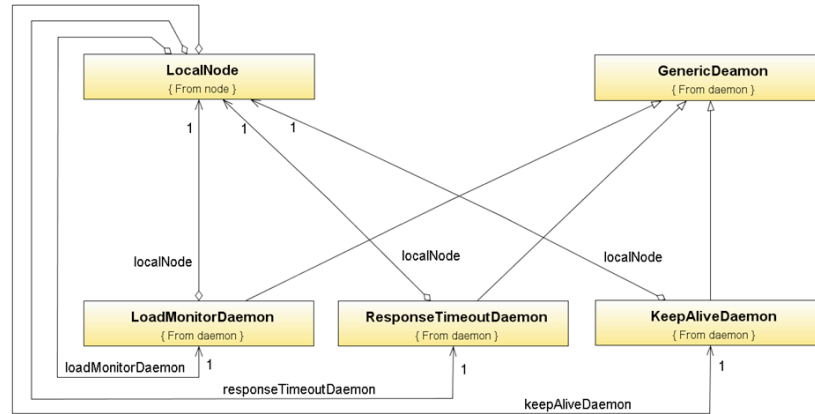


Figure 81: Collaboration between LocalNode and the Daemons

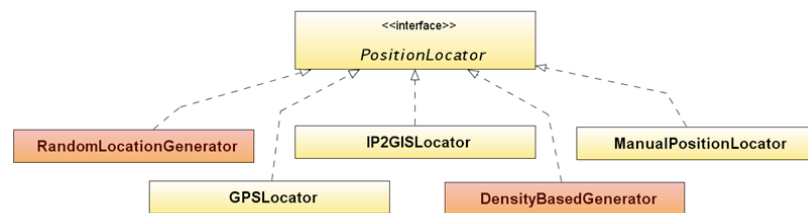


Figure 82: Supported implementations of PositionLocator - the interface used to determine the coordinates of the single peers

- *DensityBasedGenerator* - here, the coordinates are generated based on the population density given on a bitmap image (Figure 83). The algorithm for density-based generation of a peer's location has been adopted from the simulator PeerfactSim.KOM [KKM⁺07]. More densely populated areas are drawn with darker colors and the algorithm generates a peer's location in these areas with the higher possibility than the lighter, less populated areas. The final distribution of the peers is shown in Figure 84.
- *Mapping from an IP address* - the coordinates of a peer can be derived by the machine's IP-address, as described in Section 4.1.1. We query the online database of MaxMind [Max09], which is free of charge but only offers an accuracy up to the city level.
- *Using GPS (global positioning system) receivers* - if a GPS device is connected to the computer, it can be used to directly retrieve the coordinates.
- *Manually entered location* - if the location is known by the user, it can be entered manually.

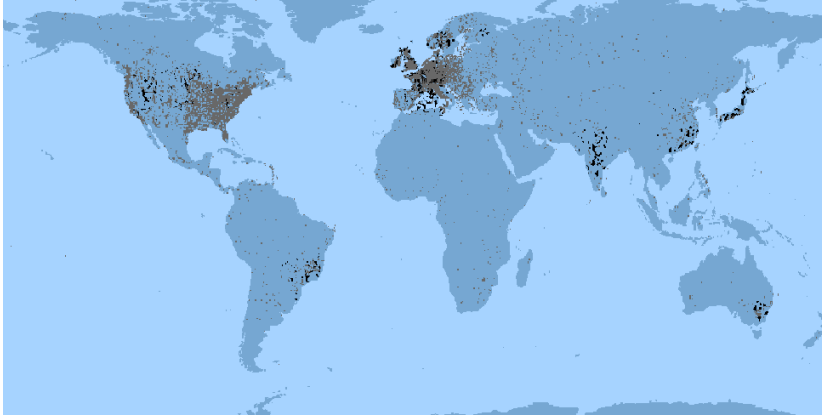


Figure 83: Sample density world map

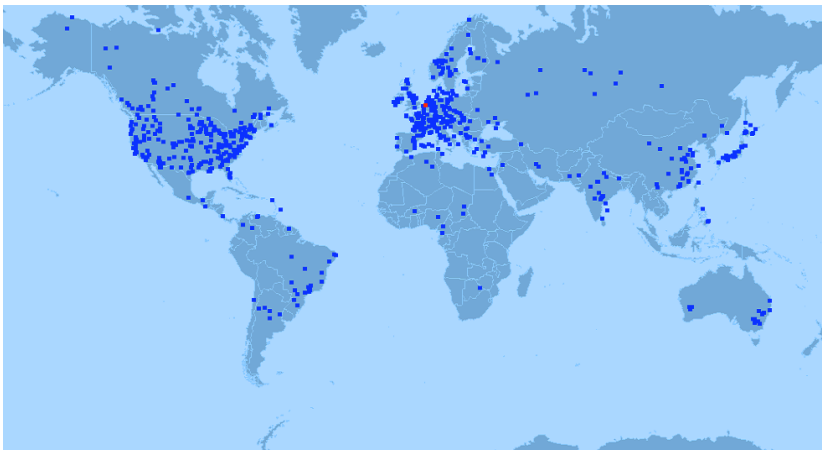


Figure 84: Resulting peer distribution

7.3 PROTOTYPE MEASUREMENTS

Prototype implementation can provide a good insight into bandwidth consumption and the response time of a prototype instance. We, therefore, ran experiments with 100 instances run on 21 computers (5 instances on run on each). We set $L_1 = 20$ and $L_2 = 40$, and the number of interconnections 10. The map of Skype users is used for distributing peers on the world map. We observed local and distant area search and finding the peer closest to a given location (referred as find closest). The experiment timeline is presented in Table 6. We captured the bandwidth consumption of every peer and the response times for queries.

The experiment results are presented in Figures 85. It shows only minor bandwidth usage (see Figure 85c) – maximum 8 KB/s has been detected on the root superpeer. The average bandwidth consumption of a superpeer was 1.2 KB/s and on a regular peer 600 B/s. The average response time for local area search was 80 ms, for distant area search 65 ms, and for find closest 40 ms. We can see that the results for local

How much bandwidth does this protocol communication consume? How fast is the response time?

Small bandwidth consumption (up to 8KB/s), very fast response time (under 80 ms)

<i>Interval</i>	<i>Operation</i>
5-25	join
30-55	local area search center in initiator's location, radius between 1km and 5000km
60-85	distant area search center 5000km away radius between 1km and 5000km
90-100	find closest - source is initiator's location or 5000km away from this node

Table 6: Timeline of experiment on prototype

area search are around 20% worse than for distant area search, which should involve more hops and more time to resolve the query. As local area search is the first operation that is run in the experiments, the interconnections are still not established, as a peer collects them from the received messages. We can see big effects of interconnections on performance. The query response time in this experiment is somewhat smaller than in the simulations (see 6). The reason for this is having small round trip times (especially in the case of communication between clients on the same machine) as all peers were in the same local network.

7.4 SUMMARY

This chapter presents a reference prototype implementation of Globase.KOM which additionally proves the viability, protocol completeness and identifies practical operational and deployment issues. It fully implements the previously described overlay 4 and simple application where a user can perform all required location-based queries (area search, finding a peer on the specific location, and finding the peer closest to a given location) and see them on the world map. Additionally, real-time statistic are provided. The amount of messages received recently and on average are displayed, with the user and system messages being presented separately. There are also two indicators of the current load of the peer, based on the two load criteria considered in the prototype – the number of peers in the zone of responsibility and the number of received messages per second.

The prototype is implemented in Java and we utilized Java RMI (Remote Method Invocation) to realize network communication. To achieve modularity, we defined three functional layers (underlay, overlay, and application), whose dependencies are kept to a minimum by communicating with the strict interfaces only. The geolocation of peers is possible in several ways: using a random generator based on the given population-density map, by mapping from an IP address, reading from a GPS receiver, or through a manually entered location.

Finally, we ran experiments with 100 instances run on 21 computers and observed the bandwidth consumption and the response times of all query types. The experiment results have shown a minor bandwidth

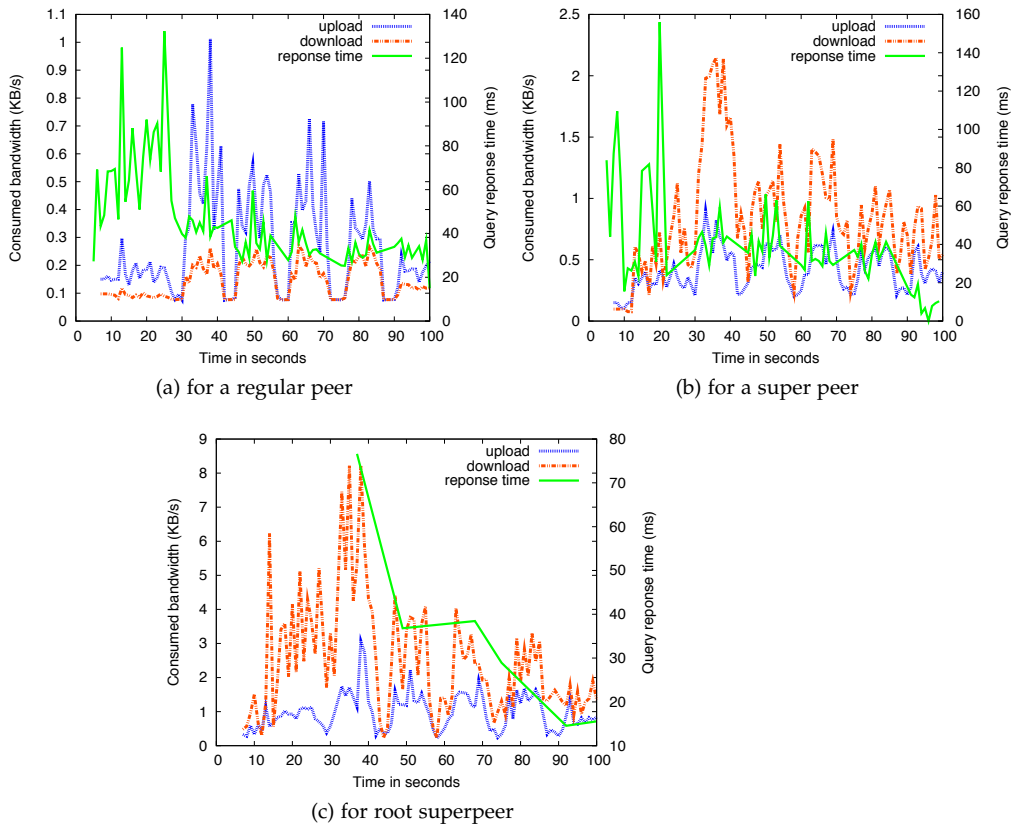


Figure 85: Bandwidth consumption and response time of a peer in Globase.KOM prototype

usage – maximum 8KB/s for the root superpeer and 900B/s for a regular peer. The average response time for area search was 70 ms and for find closest 40 ms. The query response time in this experiments is significantly smaller than in the simulations (for 100 peers, response time was 300 ms, see 6). The reason for this is having small round trip times (especially in the case of communication between clients on the same machine) as all peers were in the same local network.

A review of prototype measurements

Part IV

FINALE

CONCLUSION, SUMMARY AND OUTLOOK

This chapter concludes this thesis. In Section 8.1 we present a summary of the previous chapters and our main findings and conclusions. The implications of our findings in the networking research are given in Section 8.2. The last Section 8.3 gives an outlook on open research challenges resulting from this thesis.

8.1 SUMMARY AND CONCLUSIONS

Location-based services are very valuable nowadays and have a wide application range. In existing centrally managed solutions, however, the results of location-based search are often incomplete or outdated. A peer-to-peer approach would be available to a wide community to join and publish their services, as it operates at low costs. Each peer is responsible for the information about the object it represents, therefore the updating and publishing of information is done directly without a single point of failure.

The main goal of this thesis is to *prove the feasibility of engineering a peer-to-peer solution for fully retrievable location-based search*.

Chapter 1 illustrates the advantages of a peer-to-peer approach to location-based search and necessity for an engineering approach and evaluation methodology of peer-to-peer overlay networks. We defined the following abstract quality aspects in the scope of peer-to-peer overlay networks: validity, efficiency, fairness, scalability, stability, and robustness. Three key steps in achieving the main goal are identified. First is to acquire knowledge about existing design decision from the relevant solutions and identify their influence on quality aspects. Second is the design of our solution, based on those findings. Third is evaluation with clear methodology for addressing quality aspects and proof-of-concept.

Chapter 2 presents a classification and an overview of existing related work. Current peer-to-peer overlays for location based search are based on either existing DHT implementations or use tree-based indexing or routing. As an indexing structure, space-filling curves are mainly used to map a two dimensional space into a one dimensional curve in order to reuse DHTs that are designed for one dimensional lookups. DHT-based solutions are not fully retrievable and have inadequate mapping between logical overlay structure and underlying network. Other approaches use space partitioning trees like KD-trees or R-trees. Routing in tree-based approaches is very expensive, not greedy, and introduces many unnecessary messages. It increases the response time and protocol overhead. Additionally, loads are most often not evenly distributed among the peers.

*Motivation for
peer-to-peer
location-based search*

*Chapter 1:
requirements, the
goal, our approach*

*Chapter 2: related
work*

*Chapter 3: analyzing
possible design
decisions*

Chapter 3 analyses design decisions of structured (Chord and Kademlia), unstructured (Gia), and hybrid (Gnutella 0.6) peer-to-peer overlays. We observe six classes of design decisions according to the reference architecture model of peer-to-peer overlays from Aberer et al [LAG⁺05]. They are: choice and management of an identifier space for entities (peers and resources), mapping this entities to the identifier space, graph embedding (routing tables, neighborhood selection), routing strategy, and maintenance (failure detection and recovery). We perform comparative evaluations of quality aspects of the four aforementioned overlays. We show the significance of relation between management of identifier space and graph embedding (property of structured overlays) which enables greedy routing and increases validity and efficiency of query resolution. Very strict bind between them, however, make the overlay unstable and vulnerable to failures of the peers. Hybrid overlays, on the other side, handle churn very well. Analysis of Gia showed the importance of considering the heterogeneity of the peers into account in the design. Based on these findings, the basis for suitable design choices to realize our goal is given: it is a hybrid, structured overlay with the identifier space containing the information about offered resources and the neighbor selection based on management of identifier space.

*Chapter 4: our
solution*

The resulting overlay Globase.KOM, its algorithms and mechanisms are described in Chapter 4. It is a structured superpeer-based overlay in the form of a tree enhanced with interconnections. Superpeers are chosen from publicly reachable, static peers with more capacity, spare bandwidth, and good network connectivity. The choice and management of the identifier space are designed to support two-dimensional queries in the most suitable way. A peer identifier contains the peer's geolocation and its zone of responsibility. Management of the identifier space is based on rectangular, not overlapping zones on the world projection. Each zone is assigned to a superpeer, located inside this zone. It is responsible for all peers inside this zone. Superpeers form the tree, which is based on the subset-relation of their zones. The algorithm for forming the zones uses a clustering algorithm to define the hotspots and checking for possible overlapping with the existing zones.

*Chapter 5: evaluation
goals and
methodology*

In Chapter 5, we set the evaluation methodology containing the detailed evaluation goals, metrics, and workloads. A set of suitable metrics and workloads is assigned to the evaluations of each quality aspect. In order to model the realistic behavior of the users regarding their geographical location, we conducted experimental measurements on Skype, peer-to-peer VoIP application. We chose Skype as it is an interactive application, closer to our application scenario than the popular file-sharing applications, where users stay online until their requests are completed. As an evaluation tool we chose the simulation framework PeerfactSim.KOM, and extended it to support various geographical distribution of peers on a world map and a location-aware churn model.

In Chapter 6 we discuss the obtained evaluation results and present the retrospection on the requirements stated in Chapter 1. First we presented the calibration of the system-wide parameters: load thresh-

old parameters L_1 and L_2 , the number of interconnections, size of cache, timeouts of operations, frequency of periodical failure detection messages, and the number of missing periodical liveness information before a failure is detected. The biggest influence on the quality of the overlay (especially efficiency and fairness), make the parameters for the load threshold and number of interconnections. The evaluation of Globase.KOM regarding the quality aspects are the following:

- *Efficiency* of all query types is significantly better than the efficiency of the related solution RectNet (e.g. queries 4 times faster). The performance of lookup is better than the most referenced overlay, Chord and just around 20% worse than in Kademlia. Our solution is, however more efficient as it achieves similar or even better performance for smaller costs in terms of size of routing table (average size of routing table for Globase.KOM is 147, Chord is 161, and for Kademlia is 3131). Additionally, the underlay-awareness of Globase.KOM proved to be the biggest among all of these three reference overlays.
- Although *load-balancing* seems to be an issue due to the tree structure of the overlay, our evaluations showed that the most loaded peer is only 1,9 times more loaded than a median loaded peer. This ratio is even smaller when the peers and superpeers are considered separately, and the ratio is comparable with the flat, pure peer-to-peer overlays. Interconnections and careful selection of parameters for zone forming diminish the expected overload of superpeers in the higher tree levels.
- *Scalability* of our solution is logarithmic and is considerably better than that in the related solution, RectNet. This is due to the interconnections, choice of peer identifier space, and greedy routing.
- Our solution proved to be very *stable* under simultaneous leaving of a large portion of the peers and frequent querying. The variation of quality metric is insignificant (up to 2%) except in the case of fairness, where it becomes ca. 20% worse. Additionally, the recovery time is under one second.
- Under simultaneous failures of a large portion of peers, superpeers from the higher levels, or in the same tree branch in our solution prove to be very *robust*. The variation of quality metrics was insignificant (retrievability up to 0,35%, other metrics up to 7%), except, again the load balance ratio, which experiences an increase of 30% when 50% of the peers simultaneously fail. The recovery time is nevertheless very short.

The biggest influence on the quality of our solution was the choice of identifier space, its management, and interconnections. A peer identifier contains the information about a peer's responsibility and its location. This allows smart selection of interconnections and efficient greedy routing. Management of identifier space allows good

underlay-awareness of the overlay and therefore faster query response time. Interconnections enable bypassing of superpeers in higher levels of the tree and therefore allowed equal load distribution among the superpeers. Fast recovery time and small performance variation under extreme churn and critical failures is to be credited by the various maintenance strategies used in combination.

Chapter 7: proof of concept

Chapter 7 provides additional proof as to the viability of our solution – a reference prototype implementation Globase.KOM; it is a simple application where users can perform all required location-based queries (area search, finding a peer in the specific location, and finding the peer closest to a given location) and see the results on the world map. The geolocation of peers is possible in several ways: using random generator, based on the given population-density map, by a mapping from an IP address, reading from a GPS receiver, or by manually entering the location.

8.2 IMPLICATIONS

Enabling location-based search in a peer-to-peer fashion does not mean merely “yet another overlay”. It brings significant advantages over client-server realized location-based search as it is available to a wider community to join and publish their services and thereby overcomes the problems of existing solutions, i.e. retrieval of all up-to-date information related to an area. We show, however its broader applicability in two examples:

Implications of our work: two examples

1. We proved that enabling location-based search in peer-to-peer applications can significantly improve overlay-underlay matching and decrease the additional overlay delay. We argue that many more applications can benefit from location-awareness, especially distributed multimedia delivery.
2. Additionally, peer-to-peer collaboration applications such as peer-to-peer software development environment or massive multi-player online games can benefit from Globase.KOM.

Section 8.2.1 argues the influence of location-awareness on distributed multimedia communication and Section 8.2.2 presents some possible approaches for adopting our solution for peer-to-peer collaborative applications.

8.2.1 On Distributed Multimedia Communication

Distributed multimedia communication can benefit from location-awareness and decentralization

One of the significant implications of location-awareness in peer-to-peer systems is that it can improve the efficiency and quality of multimedia content delivery. Most distributed multimedia applications require a specific kind of network quality of service. Network quality of service (QoS) is the well-defined and controllable behavior of a network with respect to certain quantitative parameters like loss, delay, and throughput [Scho1][Heco6a]. Before analyzing where location-awareness can be

employed to improve the quality of service in the medium- to long-term future, here is a brief review of the QoS parameters on the network layer:

- *Loss*: The loss rate in today's Internet only very rarely exceeds critical levels as most backbones are overprovisioned and most voice and video encodings work quite well with a low packet loss rate. Available quality of service architectures like Diffserv [BBC⁺98] can be used to further improve the loss rate for certain types of traffic.
- *Throughput*: The throughput available for an application mainly depends on the available overall bandwidth. The available bandwidth in the Internet is roughly doubling every 10-14 months [Odlo3]. Therefore, bandwidth is growing even faster than the CPU power, which doubles every 18 months according to Moore's law [Moo65].
- *Delay*: The end-to-end delay δt experienced by the packets of a distributed multimedia application consists of:
 - the queuing delay δt_o experienced in the queues of the network routers
 - the processing delay δt_p incurred by the processing of the packet in the end-system and in each intermediate router
 - the transmission delay δt_{tr} incurred by the speed of light in the fibre optic cables (or electro-magnetic waves in copper cables) over the total end-to-end distance.

A brief review of the QoS parameters on the network layer

It is possible to optimize the queuing delay for delay-sensitive applications with existing QoS architectures like Diffserv. Additionally, as shown by Kelly [Kel99], with increasing bandwidths, the queuing delay becomes small compared to the other delay components. Therefore, in the medium to remote future, it will become more important to think about reducing the processing and transmission delay for multimedia applications.

The transmission delay δt_{tr} is given by the speed of the electro-magnetic waves in the cable c and the total end-to-end distance l : $\delta t_{tr} = c \cdot l$

As c is a constant, the transmission delay can only be improved by reducing the total distance l that a packet travels. For non-live content, l is indirectly improved best by caching or placing the content as close as possible to the customers. This process is the basic idea of the business model of companies offering CDN-solutions, e.g. Akamai [Aka]. For live content, however, this cannot be done. In this case, l is best optimized by choosing the shortest possible physical end-to-end path. At this point, knowledge about the geographical position of the involved edge systems and intermediate systems comes in handy, and as we argue next: the challenge is to create an efficient overlay for distributed multimedia systems with respect to QoS (mainly transmission delay).

Constructing the overlay requires knowledge about the delay incurred between pairs of nodes in order to have low-delay connections.

Transmission delay will become more important for multimedia communications

Caching techniques bring content closer to the users

In a distributed communication system there is no global knowledge about the overlay structure available. Knowledge about the structure could be obtained by local measurements of the delay between two nodes, e.g. using ping or traceroute. This has obvious disadvantages:

*Drawbacks of
obtaining knowledge
about overlay
structure by ping or
traceroute*

- The traffic overhead caused by measuring the delay of all nodes.
- Only the aggregated total round-trip delay at the time of measurement is retrieved. If the load situation varies, measurements have to be repeated frequently.
- Firewalls, NAT gateways, and some providers block some kinds of measurement approaches like pings and traceroutes. This can become a particular problem if the distributed multimedia system is built from end-systems (e.g. a peer-to-peer system) because normal end-systems are typically inaccessible by pings and traceroutes.

Another alternative is to use either devices that determine geographical position (e.g. GPS receivers) or IP Geolocation databases, which determine the country and the city for an IP address (see e.g. IP2Location [IP2]). Shortening the physical distance that packets travel will clearly reduce the number of IP hops necessary to reach the destination and will therefore also decrease the other critical delay component d_p . The geographical distance between two systems is, however not necessarily exactly proportional to the transmission delay between those two systems for two reasons:

*Geo-distance \approx
network distance*

- If two IP hops are operated by different Internet service providers (ISPs), packets between the two nodes have to pass an interconnection point between the networks of the two ISPs.
- Cables are not necessarily laid in a direct line between the two routers they connect. However, because of the very high costs for laying and maintaining cables, empirical results show that the distance is strongly correlated to the incurred delay (respectively cable length).

The relationship between geographical distance and network delay was exploited in research on geographical mapping systems like IP2Geo [PS01]. In order to find that relationship, Padmanabhan and Subramanian used an empirical approach based on recorded relationships between delay and location. Among other things they discovered that the ratio of linearized distance, sum of lengths of hops along the path and geographic distance, is close to 1. A. Zivani et al. [ZFdRD05] studied the correlation between network delay and geographical distance using two databases, one with poorly and one with richly connected networks. The results indicate that the correlation becomes stronger the richer the connection of the network is and that hosts in neighboring areas have similar delays. The majority of users and hosts are located in richer connected areas and therefore a strong correlation between distance and delay is valid. The final conclusion is that a decrease in the

geographical distance between two end-systems improves the quality of distributed multimedia communications.

In summary, after the investigation of different QoS parameters, we conclude that the transmission latency is a parameter that is very important for live multimedia applications and very hard to influence with traditional quality of service methods (contrary to queuing delay or loss for example). In this context, location-awareness can be used to improve the overlay structure to minimize the incurred latencies. As the geographical distance between two end-systems is strongly correlated with the transmission delay between those two nodes, knowledge of the geographical positions can help to construct the overlay in a way that minimizes the imposed delay.

The previous analysis holds for wireless networks, as they generally consist of single hop infrastructures. This final wireless hop does not have a significant effect on the delay of a global route, which has several hops more [CGLA03]. This does not, however apply on distributed multimedia communication via not widely deployed ad-hoc or mesh networks.

Location-awareness improves overlay structure and minimizes latencies

8.2.2 On Peer-to-Peer Collaborative Applications

Another important implication of this work is certainly for peer-to-peer research on collaborative systems. The examples that will be discussed here are global software development environment and networked virtual environments.

Development of most of today's software projects is organized into globally distributed developer teams in different locations around the world. There is, however no appropriate tools specifically designed for it. Developer teams usually manipulate the same artifacts and communication will most likely happen between them. This communication (including message exchange and file transfer) obviously does not need a remote server and fits naturally to peer-to-peer communication paradigm. Requirements and the system architecture of a peer-to-peer integrated software development environment are presented by Mukherjee et al. in [MKBS08].

Collaborative applications are sensitive to delay

The delay of communication and transfer in the system are crucial in the collaborative environments for software development. Using Globase.KOM as a basis for the communication can ensure that document changes are propagated fast and communication delay is minimal. Additionally, if our peer-to-peer overlay is applied on the virtual document landscape instead of the world map, developers working on the similar documents will be closer in the overlay. Virtual document landscape could be formed using algorithms for graph drawing by force-directed placement [FR91]. These algorithms draw the graph in two dimensional space iteratively, until an equilibrium of the forces is reached. In our scenario, edges represent the documents in software development (e.g. code, specification, tests) and forces between documents would represent their semantical closeness. For example, a force between a requirement document and its corresponding specification

Our solution brings collaborative parties closer

or code is strong and attractive. A force between requirement documents from two different project parts is, on the other hand, repulsive. In that way, we have a map of semantically close content rather than geographically, which makes Globase.KOM easy applicable.

*Efficient interaction
between users in
MMOGs*

The benefits our solution brings to the case of networked virtual environments is even more obvious. In virtual worlds, users interact by exchanging messages in its surrounding (so called area of interest). Decentralized storing of the states and rules (so called game logic in the case of Massively Multiplayer Online Games) in virtual worlds is the subject of several research papers [HLo4, HCCo6, DTHKo5]. The issues a peer-to-peer approach to networked virtual environments are faced with are finding the appropriate neighbors in its “visibility area” and efficient broadcasting of update messages. If our solution were to be applied here, superpeers would be coordinators of the appropriate zones in virtual worlds. All peers in the responsibility zone of a superpeer, would receive the contact lists of the peers in their area of interest. Every time a peer moves, its coordinator will send the updates of states, rules, and visible neighbors.

In summary, distributed applications, where people search for the location-related data, or collaborative environments, where delay is critical, can benefit from the presented work. Some open research challenges of providing more services to the existing and future applications will be presented in the next section.

8.3 OUTLOOK

The engineering approach to the design of peer-to-peer overlays networks opened up a number of new research challenges and questions. Many application areas of peer-to-peer communication paradigm have common needs and are faced with the similar issues (such as robustness, stability etc.). Learning from the existing overlay designs can significantly speed up the development of future peer-to-peer systems and overcome the issues that current solutions deal with. Comparative analysis and evaluation of selected overlay networks, designed for various purposes (e.g. file sharing, peer-to-peer streaming, massively multiplayer online game) or providing various query types (e.g. range queries, full-text search) with the appropriate architecture model is a huge mission the future peer-to-peer research. The first step is defining an appropriate classification of design decisions which would embrace all of the variety of applications and their structures. This would enable the creation of a general framework for rapid development of peer-to-peer applications. The affects of applications (e.g. version control, monitoring or storage) and mechanisms (e.g. NAT traversal) which build on the top of the overlays, need to be further investigated.

*A general framework
for peer-to-peer
system design*

The next crucial step is establishing benchmarking methodology, and tools for the evaluation of solutions. That includes setting exhaustive tests, which would identify the workload limits until a quality aspect of an overlay (or which would identify the limits of the overlays in terms of quality aspects). In order to provide a fair comparison between the

overlays, they must be normalized. For example, we cannot directly compare the routing efficiency of Kademlia which has 320 contacts in the routing table of a peer and Gia which has a significantly lower number of neighbors per peer. We need to set the values of overlay parameters such that the introduced costs in all overlays are the same.

As already discussed in the Section 5.5, the most appropriate tool for large-scale evaluations of peer-to-peer overlays is simulation. The abstraction of the simulation model of the overlay can, however, influence the correctness of the simulation results. The simulation results might be very different from the quality of the final product (final application). Additionally, the simulation results in the most papers in the area are not repeatable. A similar problem appears in the field of mobile ad hoc networks and is discussed in [KCC05]. It states that “less than 15% of the published MobiHoc ([Mob09]) papers are repeatable, and only 12% of the MobiHoc simulation results appear to be based on sound statistical techniques.” Establishing a commonly used benchmarking platform, which would provide realistic workloads, benchmarking tests and the possibility to combine the prototypical implementation with the simulated environment (e.g. overlay network, user behavior) would make evaluations in the peer-to-peer area easier and more reliable, and systematical.

Finally, after the design and its evaluation, there are many challenges to making an broadly deployed overlay. Transparent coupling of legacy IP applications with the overlay, without modification of the operating systems or applications is a demanding task and the main reason why many existing overlays are not widely used and accepted. Incorporating Globase.KOM into OCALA [JKK⁺06] (Overlay Convergence Architecture for Legacy Applications), would significantly speed up its deployment.

*Benchmarking of
peer-to-peer systems*

*Common evaluation
platform for
peer-to-peer systems*

*Deployment of
peer-to-peer systems*

BIBLIOGRAPHY

- [AH02] Karl Aberer and Manfred Hauswirth. An overview on peer-to-peer information systems. In *Workshop on Distributed Data and Structures (WDAS)*, 2002. (Cited on pages 49 and 161.)
- [Aka] Akamai. <http://www.akamai.com>. (Cited on pages 3 and 139.)
- [Ando6] Nate Anderson. Voip Illegal in China until 2008, March 2006. <http://arstechnica.com/old/content/2006/03/6449.ars>. (Cited on page 177.)
- [AR04] Filipe Araújo and Luís Rodrigues. GeoPeer: A Location-Aware Peer-to-Peer System. In *Proceedings of the Network Computing and Applications, Third IEEE International Symposium on (NCA)*, pages 39–46, 2004. (Cited on page 15.)
- [Arlo2] Arlington Virginia Fire Department. Arlington County After-Action Report on the Response to the September 11 Terrorist Attack on the Pentagon. http://www.arlingtonva.us/Departments/-Fire/edu/about/docs/after_report.pdf, 2002. (Cited on page 5.)
- [AS03] James Aspnes and Gauri Shah. Skip graphs. In *Proceedings of 14th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 384–393, January 2003. (Cited on page 16.)
- [AvBo9] Shah Asaduzzaman and Gregor von Bochmann. GeoP2P: An adaptive peer-to-peer overlay for efficient search and update of spatial information. *CoRR*, abs/0903.3759, 2009. (Cited on page 17.)
- [Bar01] David Barkai. *Peer-to-Peer Computing. Technologies for Sharing and Collaboration on the Net*. Intel Press, 2001. (Cited on page 161.)
- [BAS04] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: supporting scalable multi-attribute range queries. *ACM SIGCOMM Computer Communication Review*, 34(4):353–366, 2004. (Cited on page 15.)
- [BBC⁺98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. Technical report, Network Working Group, 1998. (Cited on page 139.)

- [bbc09] Web slows after Jackson's death, 2009. <http://news.bbc.co.uk/2/hi/technology/8120324.stm> accessed: 26 June 2009. (Cited on page 3.)
- [BCKO08] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 2008. (Cited on page 11.)
- [Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975. (Cited on pages 12 and 16.)
- [Ber05] Tom Berson. Skype security evaluation, October 2005. <http://www.skype.com/security/files/2005-031securityevaluation.pdf>. (Cited on page 174.)
- [BMM⁺07] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli. Revealing Skype traffic: when randomness plays with you. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 37–48, 2007. (Cited on page 174.)
- [BMM⁺08] D. Bonfiglio, M. Mellia, M. Meo, N. Ritacca, and D. Rossi. Tracking Down Skype Traffic. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, pages 261–265, 2008. (Cited on page 174.)
- [BS06] Salman A. Baset and Henning G. Schulzrinne. An analysis of the skype peer-to-peer internet telephony protocol. In *Proceedings of 25th IEEE International Conference on Computer Communications (INFOCOM)*, 2006. (Cited on pages 173 and 175.)
- [CAI] Macroscopic topology project. <http://www.caida.org/analysis/topology/macroscopic/>. (Cited on page 165.)
- [CGLA03] Marcelo M. Carvalho and J. J. Garcia-Luna-Aceves. Delay analysis of IEEE 802.11 in single-hop networks. In *Proceedings of the 11th IEEE International Conference on Network Protocols*, pages 146–155, 2003. (Cited on page 141.)
- [CKI09] Skype statistics, visited: May 2009. <http://ckipe.com/borderless>. (Cited on pages 173 and 176.)
- [CMP07] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. An experimental investigation of the congestion control used by skype voip. In *Proceedings of International Conferences on Wireless/Wired Internet Communications - WWIC*, May 2007. (Cited on page 174.)

- [CMPo8] Luca De Cicco, Saverio Mascolo, and Vittorio Palmisano. Skype video responsiveness to bandwidth variations. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pages 81–86, 2008. (Cited on page 174.)
- [CRB⁺03] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like P2P systems scalable. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 407–418, 2003. (Cited on page 38.)
- [CRR⁺05] Yatin Chawathe, Sriram Ramabhadran, Sylvia Ratnasamy, Anthony LaMarca, Scott Shenker, and Joseph Hellerstein. A case study in building layered DHT applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 97–108, 2005. (Cited on pages 14 and 197.)
- [CSMBMG07] Marc Cardenete-Suriol, Josep Mangles-Bafalluy, Álvaro Masó, and Mónica Gorricho. Characterization and comparison of Skype behavior in wired and wireless network scenarios. In *Proceedings of the Global Communications Conference (GLOBECOM)*, November 2007. (Cited on page 174.)
- [Dar05] Vasilios Darlagiannis. *Overlay Network Mechanisms for Peer-to-Peer Systems*. PhD thesis, TU Darmstadt, Germany, July 2005. (Cited on page 50.)
- [Dato8] Momentum shifting in the GPS device market. <http://dataweek.co.za/article.aspx?pkArticleId=5342&pkCategoryId=31>, 2008. (Cited on page 49.)
- [DGSK03] Shelley Zhuang Dennis, Dennis Geels, Ion Stoica, and Randy H. Katz. On failure detection algorithms in overlay networks. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2003. (Cited on page 34.)
- [DS02] Kai Fischbach Detlef Schoder. *Peer-to-Peer: Ökonomische, technische und juristische Perspektiven*, chapter Peer-to-Peer. Anwendungsbereiche und Herausforderungen, pages 3–21. Springer Verlag, Heidelberg, 2002. (Cited on page 161.)
- [DTHK05] S. Douglas, E. Tanin, A. Harwood, and S. Karunasekera. Enabling massively multi-player online gaming applications on a P2P architecture. In *Proceedings of the*

- IEEE International Conference on Information and Automation*, pages 7–12, 2005. (Cited on page 142.)
- [DZD⁺03] F. Dabek, B. Zhao, P. Druschel, J. Kubiawicz, and I. Stoica. Towards a common API for structured peer-to-peer overlays. *Lecture Notes in Computer Science*, pages 33–44, 2003. (Cited on page 23.)
- [ESR04] ESRI. *Understanding Map Projections*. 2004. (Cited on page 169.)
- [FB74] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, March 1974. (Cited on page 15.)
- [FKN80] Henry Fuchs, Zvi M. Kedem, and Bruce F. Naylor. On visible surface generation by a priori tree structures. In *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 124–133, New York, NY, USA, 1980. ACM. (Cited on page 11.)
- [FR91] Thomas M. J. Fruchterman and Edward M. Reingold. Graph drawing by force-directed placement. *Software: Practice and Experience*, 21(11):1129–1164, 1991. (Cited on page 141.)
- [Gar05] Simson L. Garfinkel. VoIP and Skype Security, December 2005. http://www.tacticaltech.org/files/tacticaltech/Skype_Security.pdf. (Cited on pages 173 and 174.)
- [GDJ06b] Saikat Guha, Neil Daswani, and Ravi Jain. An Experimental Study of the Skype Peer-to-Peer VoIP System. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2006. (Cited on pages 174, 175, 177, 178, 180, 181, 182, and 199.)
- [Geo09] Geonames. <http://www.geonames.org/>, 2009. (Cited on page 50.)
- [GGS03] Shuheng Zhou Gregory, Gregory R. Ganger, and Peter Steenkiste. Balancing locality and randomness in dhds. Technical Report Technical Report CMU-CS-03-203, Carnegie Mellon University School of Computer Science, November 2003. (Cited on page 15.)
- [GKXSo8a] Kalman Graffi, Aleksandra Kovacevic, Song Xiao, and Ralf Steinmetz. Skyeye.kom: An information management over-overlay for getting the oracle view on structured p2p systems. In *The 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS'08)*, pages 279–286, Los Alamitos, CA, USA, December 2008. IEEE, IEEE Computer Society Press. (Cited on page 22.)

- [GL96] C. Gotsman and M. Lindenbaum. On the metric properties of discrete space-filling curves. *IEEE Transactions on Image Processing*, 5(5):794–797, 1996. (Cited on page 13.)
- [GM82] H. Garcia-Molina. Elections in a Distributed Computing System. *IEEE Transactions on Computers*, C-31:48–59, 1982. (Cited on page 64.)
- [gMa] Google Maps. <http://maps.google.com/maps>. (Cited on page 4.)
- [GMa09a] Geokit. <http://geokit.rubyforge.org/>, 2009. (Cited on page 50.)
- [GMa09b] Google maps api discussion group. <http://groups.google.com/group/Google-Maps-API/web/resources-non-google-geocoders?pli=1>, 2009. (Cited on page 50.)
- [Gnu] Gnutella protocol development. <http://rfc-gnutella.sourceforge.net/src/pong-caching.html>. (Cited on page 37.)
- [goo] Google. <http://www.google.com>. (Cited on page 3.)
- [HBo8] Tobias Hoßfeld and Andreas Binzenhöfer. Analysis of Skype VoIP traffic in UMTS: End-to-end QoS and QoE measurements. *Computer Networking*, 52(3):650–666, 2008. (Cited on page 174.)
- [HCCo6] S.Y. Hu, J.F. Chen, and T.H. Chen. VON: a scalable peer-to-peer network for virtual environments. *IEEE Network*, 20(4):22–31, 2006. (Cited on page 142.)
- [HCHo9] Te-Yuan Huang, Kuan-Ta Chen, and Polly Huang. Tuning the redundancy control algorithm of skype for user satisfaction. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2009. (Cited on page 174.)
- [HD05] Manfred Hauswirth and Schahram Dustdar. Peer-to-peer: Grundlagen und architektur. *Datenbank Spektrum*, 13:5–13, 2005. (Cited on page 162.)
- [Heco6a] Oliver M. Heckmann. *The Competitive Internet Service Provider*. John Wiley & Sons, 2006. (Cited on page 138.)
- [Heco6b] Oliver Heckmann et al. Qualitätsmerkmale von Peer-to-Peer-Systemen. Technical Report KOM-TR-2006-03, Multimedia Communications Lab, TU Darmstadt, May 2006. (Cited on page 7.)
- [Heu05] D. Heutelbeck. *Distributed Space Partitioning Trees and their Application in Mobile Computing*. PhD thesis, Fernuniversität Hagen, Hagen, Germany, 2005. (Cited on pages 17 and 111.)

- [HL04] Shun-Yun Hu and Guan-Ming Liao. Scalable peer-to-peer networked virtual environment. In *Proceedings of the 3rd ACM SIGCOMM Workshop on Network and System Support for Games (NetGames)*, pages 129–133, New York, NY, USA, 2004. ACM. (Cited on page 142.)
- [HT03] Aaron Harwood and Egemen Tanin. Hashing Spatial Content over Peer-to-Peer Networks. In *Proceedings of Australian Telecommunications, Networks and Applications Conference*. ATNAC, 2003. (Cited on page 15.)
- [HT07] Octavio Herrera and Znati Taieb. Modeling churn in p2p networks. In *Proceedings of the 40th Annual Simulation Symposium (ANSS)*, pages 33–40, Washington, DC, USA, 2007. IEEE Computer Society. (Cited on pages 81 and 166.)
- [IBM08] Geographic coordinate system, October 2008. Available online at <http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.db29.doc.spatial/csb3022a.htm>, last visited on April 23rd 2009. (Cited on pages 169 and 199.)
- [iDro7] iDrive navigator, June 2007. <http://www.idrivenavigator.com>. (Cited on page 4.)
- [Inc09] Google Inc. Google maps api. <http://code.google.com/apis/maps/>, 2009. (Cited on page 50.)
- [Inta] World Internet Usage and Population Statistics. <http://www.internetworldstats.com/stats.htm>. (Cited on page 3.)
- [IP2] IP2Location - Bringing Geography to the Internet. <http://www.ip2location.com/>. (Cited on page 140.)
- [IP209] Geolocation IP Address to Country State City ISP - IP Address to Geographic Location. <http://www.ip2geo.net/>, 2009. (Cited on page 49.)
- [iTuo9] Server crashes, slow lines frustrate iPhone buyers, 2009. <http://news.cnet.com/8301-17938-105-9988807-1.html>. (Cited on page 3.)
- [Jen03] Erica Jen. Stable or robust? What's the difference? *Complexity*, 8(3), 2003. (Cited on page 8.)
- [JFr] Jfreechart. <http://www.jfree.org/jfreechart/>. (Cited on page 120.)
- [JKK⁺06] Dilip Joseph, Jayanth Kannan, Ayumu Kubota, Karthik Lakshminarayanan, Ion Stoica, and Klaus Wehrle. OCALA: an architecture for supporting legacy applications over overlays. In *Proceedings of the 3rd conference*

on *Networked Systems Design & Implementation (NSDI)*, pages 20–20, 2006. (Cited on page 143.)

- [JMF99] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999. (Cited on page 67.)
- [JMW03] Sushant Jain, Ratul Mahajan, and David Wetherall. A Study of the Performance Potential of DHT-based Overlays. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, 2003. (Cited on page 77.)
- [Joh07] Steven Berlin Johnson. Outside.in Finally Goes Outside, June 2007. <http://outside.in/blog/2007/06/29/outsidein-finally-goes-outside/>. (Cited on page 4.)
- [KaZ] Official Website of KaZaA. <http://www.kazaa.com/>. (Cited on pages 24 and 173.)
- [KCC05] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. Manet simulation studies: the incredibles. *ACM SIG-MOBILE Mobile Computing and Communications Review*, 9(4):50–61, 2005. (Cited on page 143.)
- [Kel99] F. P. Kelly. Models for a self-managed Internet. In *Royal Society Meeting*, December 1999. (Cited on page 139.)
- [KHF⁺05] Yu Kaneko, Kaname Harumoto, Shinya Fukumura, Shinji Shimojo, and Shojiro Nishio. A location-based peer-to-peer network for context-aware services in a ubiquitous environment. In *Proceedings of the Symposium on Applications and the Internet Workshops (SAINT-W)*, pages 208–211, 2005. (Cited on page 16.)
- [KKM⁺07] Aleksandra Kovacevic, Sebastian Kaune, Patrick Mukherjee, Nicolas Liebau, and Ralf Steinmetz. Benchmarking platform for peer-to-peer systems. *it - Information Technology (Methods and Applications of Informatics and Information Technology)*, 49(5):312–319, September 2007. (Cited on pages 83 and 128.)
- [KP04] S. B. Kotsiantis and P. E. Pintelas. Recent advances in clustering: A brief survey. *WSEAS Transactions on Information Science and Applications*, 1:73–81, 2004. (Cited on page 67.)
- [KPL⁺09] Sebastian Kaune, Konstantin Pussep, Christof Leng, Aleksandra Kovacevic, Gareth Tyson, and Ralf Steinmetz. Modeling the Internet Delay Space Based on Geographical Locations. In *Proceedings of the 17th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 301–310, February 2009. (Cited on page 165.)

- [KSSo9a] Verena Kantere, Spiros Skiadopoulos, and Timos Sellis. Storing and indexing spatial data in P2P systems. *IEEE Transactions on Knowledge and Data Engineering*, 21(2):287–300, 2009. (Cited on page 14.)
- [KW06] Mirko Knoll and Torben Weis. Optimizing Locality for Self-Organizing Context-based Systems. In *Proceedings of International Workshop on Self-Organizing Systems (IW-SOS)*, September 2006. (Cited on page 14.)
- [KWSW07] Mirko Knoll, Arno Wacker, Gregor Schiele, and Torben Weis. Decentralized bootstrapping in pervasive applications. In *Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW)*, pages 589–592, 2007. (Cited on page 65.)
- [LAG⁺05] Karl Aberer Luc, Luc Onana Alima, Ali Ghodsi, Sarunas Girdzijauskas, Seif Haridi, and Manfred Hauswirth. The essence of p2p: A reference architecture for overlay networks. In *Proceedings of the 5th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pages 11–20, 2005. (Cited on pages 21, 26, 29, 35, 47, 52, 68, and 136.)
- [LCC⁺05b] Anthony Lamarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, Jason Tabert, Pauline Powledge, Gaetano Borriello, and Bill Schilit. Place lab: Device positioning using radio beacons in the wild. In *Proceedings of the 3rd International Conference on Pervasive Computing (PERVASIVE)*, May 2005. (Cited on pages 14 and 50.)
- [LHHS05] Nicolas Liebau, Oliver Heckmann, Ian Hubbertz, and Ralf Steinmetz. A Peer-to-Peer Webcam Network. In *Proceedings of Peer-to-Peer Systems and Applications Workshop associated with KiVS'05*, pages 151–154, March 2005. (Cited on page 4.)
- [Lim] Official Website of LimeWire. <http://www.limewire.com/>. (Cited on page 95.)
- [LK00] Jonathan K. Lawder and Peter J. H. King. Using space-filling curves for multi-dimensional indexing. In *Proceedings of the 17th British National Conference on Databases (BNCOD)*, pages 20–35, London, UK, 2000. Springer-Verlag. (Cited on page 13.)
- [Mat09] Mathworks: Map projections reference, 2009. Available online at <http://www.mathworks.com>, last visited on April 23rd 2009. (Cited on pages 170 and 199.)

- [Max09] GeoIP mapping IP addresses to geographical locations. <http://www.maxmind.com>, 2009. (Cited on pages 49, 50, 128, and 174.)
- [MJ05] Dillip Mohapatra and Suma SB. Jataayu. Survey of location based wireless services. In *Proceedings of IEEE International Conference on Personal Wireless Communications (ICPWC)*, pages 358–362, 2005. (Cited on page 4.)
- [MKBS08] Patrick Mukherjee, Aleksandra Kovacevic, Michael Benz, and Andy Schürr. Towards a peer-to-peer based global software development environment. In *Proceedings of the Software Engineering Conference SE2008*, pages 204–216, February 2008. (Cited on page 141.)
- [MLK04] Anirban Mondal, Yi Lifu, and Masaru Kitsuregawa. P2PR-Tree: An R-Tree-Based Spatial Index for Peer-to-Peer Environments. In *Current Trends in Database Technology - EDBT 2004 Workshops*, pages 516–525, 2004. (Cited on page 16.)
- [MM02] Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the xor metric. In *IPTPS '01: Revised Papers from the First International Workshop on Peer-to-Peer Systems*, pages 53–65, 2002. (Cited on pages 23, 58, 99, and 116.)
- [Mob09] The ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2009. <http://www.sigmobile.org/mobihoc/>. (Cited on page 143.)
- [Moo65] Gordon E. Moore. Cramming more components onto integrated circuits. In *IEEE Electronics*, volume 38, April 1965. (Cited on page 139.)
- [MS07] Peter Mahlmann and Chistian Schindelbauer. *Peer-to-Peer Netzwerke – Algorithm und Methoden*. Springer, 2007. (Cited on page 163.)
- [MSN09] Msn encarta: Online encyclopedia, dictionary, atlas, and homework, 2009. Available online at <http://encarta.msn.com/>, last visited on April 23rd 2009. (Cited on pages 170 and 199.)
- [Nob09] Leo Nobach. Assessing qualities of peer-to-peer search overlays, 2009. Bachelor thesis, TU Darmstadt, advisor: Aleksandra Kovacevic. (Cited on page 26.)
- [ns209] The network simulator – ns-2, 2009. <http://www.isi.edu/nsnam/ns/>. (Cited on page 9.)
- [Odlo3] Andrew M. Odlyzko. Internet traffic growth: Sources and implications. In *Proceedings of SPIE Conference on Optical Transmission Systems and Equipment for WDM*

- Networking II*, pages 1–15, September 2003. (Cited on page 139.)
- [Ope] Openmap - open systems mapping technology. <http://openmap.bbn.com/>. (Cited on page 120.)
- [O’R05] Tim O’Reilly. What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software, 2005. <http://oreilly.com/web2/archive/what-is-web-20.html>. (Cited on page 3.)
- [Out] Outside.in. <http://www.outsidein.com>. (Cited on page 4.)
- [Pee] PeerfactSim.KOM: A Simulator for Large-Scale Peer-to-Peer Networks. <http://www.peerfactsim.com>. (Cited on pages 83 and 165.)
- [Pin] The PingER Project. <http://www-iepm.slac.stanford.edu/pinger/>. (Cited on page 165.)
- [PKL⁺08] Konstantin Pussep, Sebastian Kaune, Christof Leng, Aleksandra Kovacevic, and Ralf Steinmetz. Impact of user behavior modeling on evaluation of peer-to-peer systems. Technical Report KOM-TR-2008-07, November 2008. (Cited on pages 78, 80, and 198.)
- [Pla] Place Lab. <http://www.placelab.org/>. (Cited on page 50.)
- [Pla06] An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services, 2006. www.planetlab.com. (Cited on page 83.)
- [PS01] Venkata N. Padmanabhan and Lakshminarayanan Subramanian. An Investigation of Geographic Mapping Techniques for Internet Hosts. *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, page 13, 2001. (Cited on pages 49 and 140.)
- [qua09] QuaP2P Project Web Site, 2009. <http://www.quap2p.tu-darmstadt.de/>. (Cited on page 7.)
- [RBR⁺04] Mema Roussopoulos, Mary Baker, David S. H. Rosenthal, TJ Giuli, Petros Maniatis, and Jeff Mogul. 2 P2P or Not 2 P2P? In *Proceedings of the 3rd International Workshop on Peer-to-Peer Systems (IPTPS)*, volume 3279 of LNCS, pages 33–43. Springer, 2004. (Cited on page 161.)
- [RD01] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*,

- pages 329–350, Heidelberg, Germany, November 2001. (Cited on pages 14 and 161.)
- [RFH⁺01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 161–172, 2001. (Cited on pages 15 and 16.)
- [RKY⁺02] Sylvia Ratnasamy, Brad Karp, Li Yin, Fang Yu, Deborah Estrin, Ramesh Govindan, and Scott Shenker. GHT: A Geographic Hash Table for Data-Centric Storage. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, September 2002. (Cited on page 15.)
- [RMO6] John Risson and Tim Moors. Survey of research towards robust peer-to-peer networks: search methods. *Computer Networks*, 50(17):3485–3521, 2006. (Cited on page 161.)
- [RMI] Remote method invocation home. <http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp>. (Cited on page 120.)
- [RMMo8] Dario Rossi, Marco Mellia, and Michela Meo. A Detailed Measurement of Skype Network Traffic. In *Proceedings of the 7th International Workshop on P2P Systems (IPTPS)*, February 2008. (Cited on page 174.)
- [RSW05] Michael K. Reiter, Asad Samar, and Chenxi Wang. Distributed Construction of a Fault-Tolerant Network from a Tree. In *Proceedings of the 24th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 155–165, 2005. (Cited on page 57.)
- [Scho1] Jens Burkhard Schmitt. *Heterogeneous Network Quality of Service Systems*. Kluwer Academic Publishers, June 2001. ISBN 0-793-7410-X. (Cited on page 138.)
- [Scho8] James L. Schmidhammer. Agglomerative hierarchical clustering methods, 2008. Course material, available online at <http://www.bus.utk.edu/stat/Stat579/HierarchicalClusteringMethods.pdf>, last visited on September 7th 2008. (Cited on page 66.)
- [SE05] Ralf Steinmetz and Klaus Wehrle (Edits.). *Peer-to-Peer Systems and Applications*. Springer, September 2005. (Cited on page 162.)
- [SENBo7] Moritz Steiner, Taoufik En-Najjary, and Ernst W. Bier-sack. A global view of KAD. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*

- (IMC), pages 117–122, 2007. (Cited on pages 173 and 177.)
- [SFKTo6] Kyoungwon Suh, Daniel R. Figueiredo, Jim Kurose, and Don Towsley. Characterizing and detecting skype-relayed traffic. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2006. (Cited on page 174.)
- [Shio1] Clay Shirky. *Peer to Peer: Harnessing the Power of Disruptive Technologies*, chapter Listening to Napster, pages 19–28. O'Reilly & Associates, 2001. (Cited on page 161.)
- [Skro7] Rich Skrenta. Why Physical Location Matters in Online News and Community. O'Reilly Where2.0 Conference, March 2007. http://conferences.oreillynet.com/cs/where2007/view/e_sess/12543. (Cited on page 4.)
- [Skyb] Official Website of Skype. <http://www.skype.com>. (Cited on pages 24 and 95.)
- [Sli] Slifter. <http://www.slifter.com>. (Cited on page 4.)
- [SLLYo8] Wei Song, Ruixuan Li, Zhengding Lu, and Guangcan Yu. FAN: A Scalable Flabellate P2P Overlay Supporting Multi-Dimensional Attributes. In *Proceedings of 22nd International Conference on Advanced Information Networking and Applications (AINA)*, pages 1005–1012, 2008. (Cited on page 14.)
- [SMo9] Hendrik Schulze and Klaus Mochalski. Internet study 2008/2009, 2009. <http://www.ipoque.com/study/ipoque-Internet-Study-08-09.pdf>. (Cited on page 4.)
- [SMK⁺o1] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 149–160, 2001. (Cited on pages 14, 16, and 23.)
- [SMLN⁺o3] Ion Stoica, Robert Morris, David Liben-Nowell, David Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications. *IEEE Transactions on Networking*, 11(1):17–32, 2003. (Cited on pages 99 and 116.)
- [SPBPo6] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using PlanetLab for Network Research: Myths, Realities, and Best Practices. *ACM SIGOPS Operating Systems Review*, 40(1):17–24, 2006. (Cited on page 83.)

- [SR06] Daniel Stutzbach and Reza Rejaie. Understanding Churn in Peer-to-Peer Networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 189–202, 2006. (Cited on page 80.)
- [SW04] Ralf Steinmetz and Klaus Wehrle. Peer-to-peer networking and computing. *Informatik Spektrum*, 27(1):51–54, February 2004. (Cited on page 162.)
- [TCP08] A packet sniffer. <http://www.tcpdump.org>, 2008. (Cited on page 175.)
- [THS⁺04] Egemen Tanin, Aaron Harwood, Hanan Samet, Sarana Nutanong, and Minh Tri Truong. A serverless 3D world. In *Proceedings of the 12th Annual ACM International Workshop on Geographic Information Systems (GIS)*, pages 157–165, 2004. (Cited on page 15.)
- [TM09] IP2Location TM. Bringing geography to the internet. <http://www.ip2location.com/>, 2009. (Cited on page 49.)
- [WHO85] Nicname/whois. IETF, RFC 812, October 1985. (Cited on page 49.)
- [WSS05] Bernard Wong, Aleksandrs Slivkins, and Emin Gun Sirer. Meridian: a lightweight network location service without virtual coordinates. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 85–96, 2005. (Cited on page 15.)
- [WZK05] Haojun Wang, Roger Zimmermann, and Wei-Shinn Ku. ASPEN: an adaptive spatial peer-to-peer network. In *Proceedings of the 13th annual ACM International Workshop on Geographic Information Systems (GIS)*, pages 230–239, 2005. (Cited on page 16.)
- [XY07] Haiyong Xie and Yang Richard Yang. A Measurement-based Study of the Skype Peer-to-Peer VoIP Performance. In *In Proceedings of 6th International Workshop on Peer-to-Peer Systems (IPTPS)*, February 2007. (Cited on pages 174, 175, 176, and 180.)
- [YGM02] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS)*, page 5, 2002. (Cited on page 161.)
- [ZCWQ08] Sheng-Hui Zhao, Gui-Lin Chen, Guo-Xin Wu, and Ning Qian. A strategy for Selecting Super-Peer in P2P and Grid Based Hybrid System. In *Proceedings of the 3rd International Conference on Technologies for E-Learning and*

- Digital Entertainment (Edutainment)*, pages 192–199, 2008. (Cited on page 50.)
- [ZFdRD05] Artur Ziviani, Serge Fdida, José F. de Rezende, and Otto Carlos Duarte. Improving the accuracy of measurement-based geographic location of internet hosts. *Computer Networks and ISDN Systems*, 47(4):503–523, 2005. (Cited on page 140.)
- [ZGS03] Shuheng Zhou, Gregory Ganger, and Peter Steenkiste. Location-based Node IDs: Enabling Explicit Locality in DHTs. Technical Report CMU-CS-03-171, Carnegie Mellon University, September 2003. (Cited on page 15.)
- [ZK04] Chi Zhang and Arvind Krishnamurthy. SkipIndex: Towards a scalable peer-to-peer index service for high dimensional data. Technical Report TR-703-04, Princeton University, 2004. (Cited on page 16.)
- [ZKW04] Roger Zimmermann, Wei-Shinn Ku, and Haojun Wang. Spatial Data Query Support in Peer-to-Peer Systems. In *Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts - (COMPSAC)*, pages 82–85, 2004. (Cited on page 16.)
- [ZKW05] Chi Zhang, Arvind Krishnamurthy, and Randolph Y. Wang. Brushwood: Distributed Trees in Peer-to-Peer Systems. In *Proceedings of the 4th International Workshop on Peer-To-Peer Systems (IPTPS)*, pages 47–57, February 2005. (Cited on page 16.)
- [ZSLSo6] Changxi Zheng, Guobin Shen, Shipeng Li, and Scott Shenker. Distributed Segment Tree: Support of Range Query and Cover Query over DHT. In *Proceedings of the 5th International Workshop on Peer-to-Peer Systems (IPTPS)*, 2006. (Cited on page 14.)

Part V

APPENDIX

PEER-TO-PEER is a decentralized communication paradigm in the Internet. The term *peer-to-peer* was created in the year 2000. According to an initial definition given by Clay Shirky in [Shio1] peer-to-peer is “a class of applications that takes advantage of resources - storage, cycles, content, human presence - available at the edges of the Internet. Because accessing these decentralized resources means operating in an environment of unstable connectivity and unpredictable IP addresses, peer-to-peer nodes must operate outside the DNS and have significant or total autonomy from central servers.” Shirky also provides a litmus test, according to which, an application is peer-to-peer if (1) it allows for variable connectivity and temporary network addresses and (2) it gives the nodes at the edges of the network significant autonomy.

The IRTF Peer-to-Peer Research Group applies two definitions:

“... a way of structuring distributed applications such that the individual nodes have symmetric roles. Rather than being divided into clients and servers each with quite distinct roles (such as Web clients vs. Web servers), in peer-to-peer applications a node may act as both a client and a server. Peer-to-peer systems are, in general, deployable in an ad-hoc fashion, without requiring centralized management or control. They can be highly autonomous, and can lend themselves to anonymity.”

Further, the group applies [RM06] as its overview paper, where a peer-to-peer network is defined to exhibit three characteristics self organization, symmetric communication, and distributed control. This definition is taken from [RBR⁺04]. Further, a self organizing peer-to-peer network is defined according to [RD01] as “a network that automatically adapts to the arrival, departure, and failure of nodes.”

The peer-to-peer working group of the Internet2 consortium defined peer-to-peer computing as “... sharing of computer resources and services by direct exchange between systems. These resources and services include the exchange of information, processing cycles, cache storage, and disk storage for files” (see [DS02]). The working groups seems to have become inactive after April 2004. In a similar way Barkai [Bar01] defined peer-to-peer computing as “... a network based computing model for applications where computers share resources via direct exchanges between participating computers.”

In research, several people have also defined peer-to-peer systems. Yang and Garcia-Molina [YGM02] define peer-to-peer systems as: “Peer-to-peer (P2P) systems are distributed systems in which nodes of equal roles and capabilities exchange information and services directly with each other.” Aberer and Hauswirth [AHO2] extract from Shirky’s definition three underlying principles: the principle of sharing resources, the principle of decentralization, and the principle of self organization.

In a later work, Hauswirth [HD05] modifies this definition by ascribing the following characteristics to peer-to-peer systems: symmetry of roles, decentralization (no central coordination, no central data base, no peer having a global view of the system), self organization, autonomy of peers, unreliability of peers and network links, availability of data stored in the system in the presence of distributed storage, unreliability, unknown trustworthiness of peers, etc.

In summarizing all of these characteristics, the most complete definition to be found was composed by Steinmetz and Wehrle in [SW04, SE05] where they summarize nine important characteristics of peer-to-peer systems.

A peer-to-peer system is a self-organizing system of equal, autonomous entities (peers) which aims for the shared usage of distributed resources in a networked environment avoiding central services.

They further define 9 properties that characterize peer-to-peer systems, which they group in to the two categories decentralized resources usage and decentralized self-organization (though a single system rarely exhibits all of these properties):

1. Resources of interest (bandwidth, storage, processing power) are used in a manner as equally distributed as possible and are located at the edges of the network, close to the peers.
2. Within a set of peers, each utilizes the resources provided by other peers. The most prominent examples for such resources are storage and processing capacity. Other possible resources are connectivity, human presence, or geographic proximity.
3. Peers are interconnected through a network and, in most cases, distributed globally.
4. In order to deal with variable connectivity of peers peer-to-peer systems, are introduced to introduce new address and name spaces above the traditional Internet address level. Hence, content is usually addressed through unstructured identifiers, derived from the content with a hash function. Consequently, data is no longer addressed by location (the address of the server) but by the data itself.
5. In order to utilize shared resources, peers interact directly with each other. In general, this interaction is achieved without any central control or coordination.
6. Peers directly access and exchange the shared resources they utilize, without a centralized service.
7. In a peer-to-peer system, peers can act both as clients and servers.
8. Peers are equal partners with symmetric functionality. Each peer is fully autonomous regarding its respective resources.
9. Ideally, resources can be located without any central entity or service.

PEER-TO-PEER OVERLAY NETWORK VS. PEER-TO-PEER SYSTEM A Peer-to-Peer Overlay Network (or just Peer-to-Peer Overlay) is a logical network, which is built on top of the IP network. Peers are connected by virtual or logical links, each corresponding to one or more physical links.

The term “peer-to-peer network” is often used. Mahlmann and Schindelhauer [MS07] state that the research community has agreed roughly that a peer-to-peer network is an overlay network where the participating computers communicate as peers without central coordination in order to jointly provide an application.

We observe peer-to-peer overlay networks and peer-to-peer systems separately. An overlay network provides the basic functionality of indexing and storing, and with the application on top, it builds a peer-to-peer system. In this thesis, we refer to “peer-to-peer overlay network” also as “peer-to-peer network”, “overlay” and “overlay network”. A peer-to-peer system, on the other hand includes the peer-to-peer overlay and an application on top of it.

SUPPLEMENTARY DETAILS TO CHAPTER 3: SIMULATION SETTINGS

For our simulations, we use PeerfactSim.KOM [Pee] with the GNP **network layer model** [KPL⁺09]. This is based on *Internet End-to-End-Delay* measurements of the CAIDA [CAI] and the PingER [Pin] project. The model assigns a geographical position to each peer and defines the Minimum Roundtrip Time (minRTT) based on the Euclidean distance between two peers. The delay of a link is then calculated by adding a random part, the *jitter*, to the half of the minimum roundtrip time previously determined.

$$\text{delay} = \text{jitter} \cdot \frac{\text{minRTT}}{2}$$

Additionally, the network layer delays and queues packets depending on their size and the available bandwidth, modeling a maximum up- and downstream bandwidth at each peer. For the demand of the simulations to be realistic, different up- and download bandwidth capacities are assigned to peers randomly.

We define a set of **resources** with a unified identifier (UID) assigned and attached to each of them (*rank*). For publishing, the peers choose resources from a global set, modeling applications on top. We define the *popularity* of a document as the document's *relative frequency to be stored and looked up*.

We choose the Zipfian (or Zeta) distribution for document popularity. The Zipf distribution can be observed in many discrete distributions, the most popular example is the frequency of each word in languages.

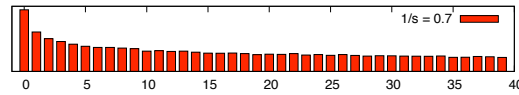


Figure 86: A set of 40 documents with their relative frequency, Zipf-distributed

Several times during the simulation, two random keys are chosen out of this set and their frequencies are swapped. These time points are equally distributed with a mean interval of 5 hours divided by the document count, so that after at least 5 hours most ranks have been swapped once. The goal of this procedure is to measure how the overlays react to sudden changes of the popularity.

In order to conduct comparable simulations of four different overlays, we must define the **common operations**:

- **Join()**: A peer connects to an overlay network.
- **Leave()**: A peer leaves an overlay network.
- **Publish(Set of resources)**: A peer makes the resources available to the public, which are defined in the set, given as a parameter. In

the DHTs, this is done by publishing every single resource of this set separately. For this, the rank of each resource is hashed and used as the DHT key. The command also initiates republishing after a certain period. In Gnutella, the documents are only made available locally. If the peer is a leaf, the connected ultrapeers are additionally informed about the change of the peer's shared resource set.

- **Search(Rank):** The node requests at least one resource with a given rank identifier. In the DHT overlays, this is done by hashing the rank and looking it up in the given overlay. In Gnutella and Gia, a query is started containing the rank to lookup. A query hit occurs, if a published rank matches the given rank in the query exactly. If at least one hit is received, the query is finished.

The only operation on the overlays which is carried out by each peer is the periodical search for keys previously stored. In the DHT overlays, the operations are extended by periodical republish operations. In these evaluations, we do *not* assign different query frequencies to the users. Instead, every user randomly queries for a document, at equally distributed times after the initiation of the queries. All users have the same mean interval of 10 seconds.

We used an exponential **churn** model, proposed by Herrera et al. [HT07] with the following parameters:

Peer type	Ratio (%)	Conn. Factor	Relative session interval
Benefactor	10	75	100
Peer	40	50	10
Peeper	50	45	1

Table 7: Churn Model Parameters

Our experiment timeline is presented in Figure 87. At the beginning, one peer joins the network and the peer count is increased by the factor $10^{0.5}$ every ten minutes. This join procedure is carried out until the desired peer count is reached. In a scenario with 10 000 peers, this is after approximately 80 minutes. The exponential growth of the peers, the jining process becomes faster, while maintaining stability in the early phases, unlike linear growth.

At 100 minutes, the peers start to publish documents at equally distributed intervals across 20 minutes, as a sudden publish could congest the network. The document count of each peer is exponentially distributed with a mean of 20 documents, and is drawn from the defined Zipfian or unique document set.

After 120 minutes, the peers begin lookups at equally distributed intervals with a mean interval of 10 minutes. After 120 minutes, the churn process starts and reaches the steady state after another 60 minutes. At 180 minutes, measurements begin and are taken in every 20 minutes until the simulation ends.

Overlay parameters are set as following:

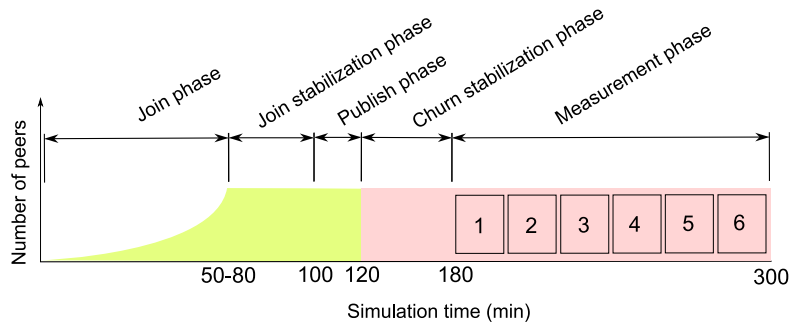


Figure 87: Experiment Timeline

<i>Parameter</i>	<i>Value</i>
Identifier size	180
Fingertable refreshing interval (fixFingerInterval)	10s
Stabilization interval (stabilizeInterval)	10s
Predecessor refreshing interval (checkPreInterval)	10s

Table 8: Chord: parameter settings

<i>Parameter</i>	<i>Value</i>
Identifier length	160
Order of the routing tree	2
Bucket size (k)	20
Size of the replacement cache	12
Number of unsuccessful contacting attempts allowed	1
Bucket entry ping interval	1.2
Number of initial contacts	30
Concurrent lookups (α)	3
Timeout for unresponsive nodes	0.5 seconds
Lookup timeout	10 seconds
DHT value size	13 bytes
Expiration time of DHT values	3.2 hours
Republish interval	0.2 hours

Table 9: Kademlia: parameter settings

<i>Parameter</i>	<i>Value</i>	<i>Capacity</i>	<i>Abs. freq.</i>
Max. TTL for ping messages	3	10000	1%
Max. query TTL	512	1000	5%
Max. replies for ending a query	1	100	29%
Ping interval	20s	10	55%
Timeout for contact communication	8s	1	20%
Query timeout	5s		
Neighbor ping interval	0.6s		
Identifier size	11 bits		
Timeout for pong route entry	5s		
Timeout for pong accept entry	5s		

Table 10: Gia: parameter settings

<i>Parameter</i>	<i>Value</i>
Timeout for connection attempt	1s
Max. number of leaf to ultrapeers (UPs)	3
Max. connections of an UP to leaves	30
Max. connections of an UP to another UP	32
Number of entries in the pong cache	5
Connection attempts before stale	1
Size of the list of UPs advertised on a connection attempt	10
Interval of ping messages	15s
Timeout for a ping message	1s
TRY_ULTRAPEERS after amount of connections	12
Interval after unsuccessful bootstrapping repeats	5s
Timeout for requests other than pings or conn. attempts	1s
Depth of deeper dynamic queries (DQ)	2
Duration of each controlled broadcast step (DQ)	1s
Duration of the probe query step (DQ)	1s
Number of hits to stop further searching (DQ)	1
Query timeout for a leaf	40s
Time to remember queries already relayed	30s
Peers above this bandwidth may be ultrapeer	1kbps
Peers above this bandwidth must be ultrapeer	2kbps
Ultrapeer ratio of potential ultrapeers	90%

Table 11: Gnutella 0.6: parameter settings

MAP PROJECTIONS

The description of geolocation is in the form of *longitude* and *latitude* values in DMS format, which is standard in geographic information systems (GIS) and Global Positioning Systems (GPS) (e.g. $44^{\circ}49'14''\text{N}$ $20^{\circ}27'44''\text{E}$ for Belgrade). It is described in the geographical coordinate system (see Figure 88), as a three dimensional coordinate system with the center in the earth's center, one plane containing the equator, the other plane orthogonal to it (containing lines of longitude or meridians). The longitude and latitude are angles between the earth's center and the observed point on the surface [ESRo4]. The longitude is the angle between the projection of the point to the equator plane and prime meridian passing through Greenwich, England and range from -180° west to $+180^{\circ}$ east. Latitude is the angle on the plane orthogonal to the equator, containing an observed point, measured from the equator, ranging from -90° at the South Pole to $+90^{\circ}$ at the North Pole.

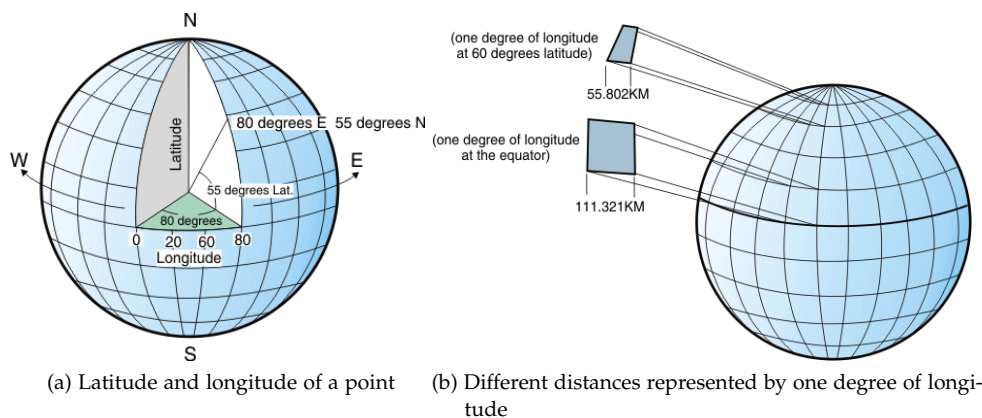


Figure 88: Representation of a point on the earth's surface in the Geographical Coordinate System (Source: [IBMo8])

The latitude-longitude description in the geographical coordinate system allows exact, unique addressing of all points on the earth's surface. However, this description is not equidistant, meaning that the distance represented by one degree varies, depending on its position. For example, one degree of longitude at the equator is 111 321 km, while at 60° latitude it is only 55 802 km [ESRo4]. Accurate distance or area measurement and representation of geodata on two dimensional maps will be the subject of the following.

The earth is neither a perfect sphere nor a perfect spheroid due to gravitational and surface feature variations (for example, the South Pole is closer to the equator than the North Pole). Representing the earth's surface on two-dimensional mediums (e.g. paper or computer

screen) and measuring distances and areas requires mathematical transformation referred to as a *map projection*. All map projections introduce some kind of distortion because a spheroid cannot be mapped to a plane without stretching, tearing, or shrinking. The aim of each of them differs, e.g., the representation of a particular area in large scale, or the whole world in small scale. The usual transformation method is systematical projection of locations on the earth's surface on developable surfaces such as cones, cylinders, or planes. Many common map projections can therefore be classified as conical, cylindrical, or planar. However, not all projections fit into these groups. There are also special groups such as circular or star.

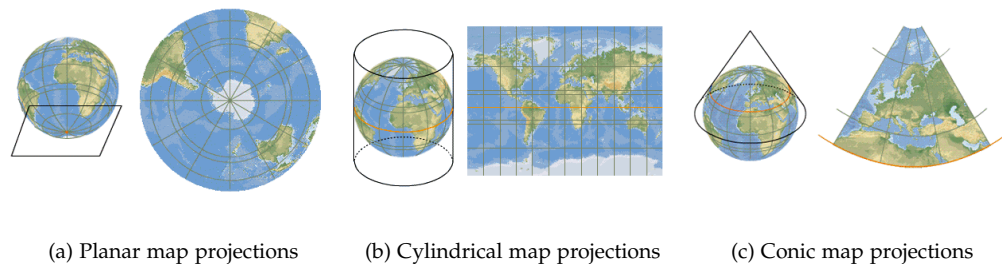


Figure 89: Three types of map projections (Source: [MSN09])

For the needs of location-based search, described by the functional requirements in 1.2.2, a map projection must fulfill all of the following requirements: simple depiction, precise on small areas (e.g. on city level), and simple calculation. Plate Carée projection is a simple cylindrical projection, useful for index maps. This projection plots latitude-longitude points directly on a regular X, Y graph, assuming the Earth is a sphere. The lines of longitude on the graph are spaced using the same scale as the latitude lines, forming a grid of equal sized rectangles. Figure 90 shows a world map using a Plate Carée projection.

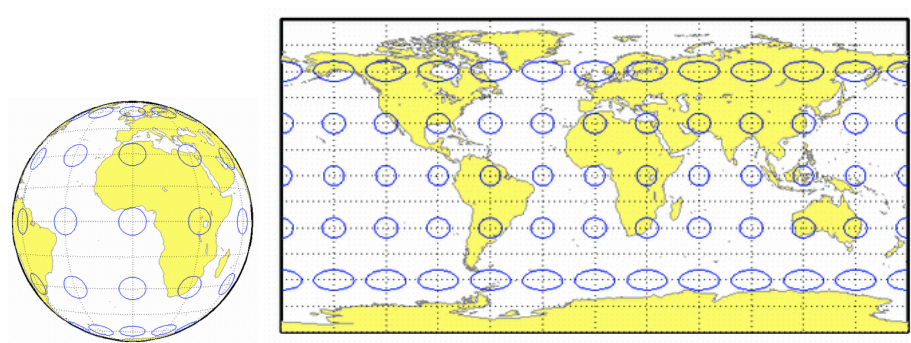


Figure 90: Plate Carée projection (Source: [Mat09])

The distortion introduced by the Plate Carée increases with the latitude. For zones lying on the equator there is little distortion, but

zones far away from it are strongly distorted. This distortion must be taken into account when performing geographical calculations. If one was to search for zones lying within a specified radius of a point on the earth's surface, this circle is transformed into an ellipse on the overlay's flat projection. To visualize this, the mapping of the same circle on different locations (Tissot's indicatrix) is shown in Figure [90](#).

GEOGRAPHICAL DISTRIBUTION OF PEERS AND LOCATION-AWARE CHURN MODEL

As the geographical location of the participating peers affects the structure of the Globase.KOM overlay, the relation between the online behavior of the users and their location is of special interest. Most of the aforementioned measurement studies did not investigate this relation. Steiner et al. have partly addressed this issue in [SENB07]. They showed the geographic distribution of the peers in the Kad network and the number of peers online during a particular time of day in the top five countries, revealing a visible diurnal pattern.

In file sharing applications, users stay online until their requests are completed. In interactive applications, such as VoIP, web or mail, users are usually online during working hours and offline during the night. Additionally, the geographical distribution of the peers is also different. For example, the most users of Kad network are from China while the majority of the Skype users are in Europe. Location-based search implies interactivity which means the measurement studies of file-sharing applications and corresponding churn and geographical distribution models are not compatible with our solution. Capturing the user behavior in a popular interactive peer-to-peer application is a necessary step for deriving a realistic geographical distribution of peers and location-aware churn.

Skype is the most popular peer-to-peer application, counting 442 million usernames, 42 million users active daily and more than 17 million users online at the same time [CKI09]. Despite its popularity, there is very little information available regarding its protocol and network structure. What is known is that it is a hybrid, superpeer-based network with similarities to KaZaA [KaZ], as both companies share the same founders and are likely to employ the same technical staff [Gar05].

Figure 91 presents the structure of the Skype network. *Supernodes* are connected to each other, maintaining an overlay network while each *ordinary node* is connected to one or two elected supernodes. Ordinary nodes get promoted to supernodes if there is enough spare bandwidth and they are publicly reachable. Its hybrid structure supports heterogeneity of the peers like Globase.KOM does, making it additionally suitable for our workload model.

D.1 EXISTING MEASUREMENTS STUDIES OF SKYPE

The first measurement study of Skype [BS06] gave some insight into the protocol explaining the login phase, NAT and firewall traversal, association to the supernodes, user search, call establishment, media transfer, codecs and conferencing under different network se-

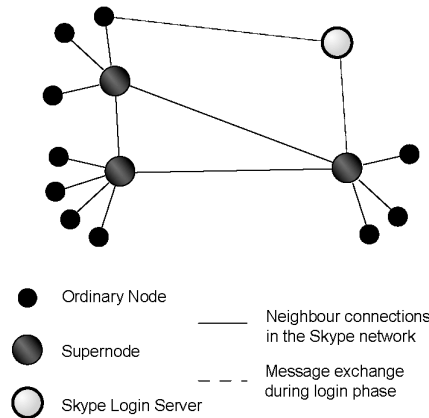


Figure 91: Skype Network

tups. Following studies focused on traffic analysis [HBo8, RMMo8, BMM⁺07, BMM⁺08], congestion control [CMP07, CMP08], relayed traffic [SFKT06], user satisfaction [HCH09, CSMBMG07], and security issues [Gar05, Bero5].

Guha et al. ran a measurement study of Skype over a five month period, from September 2005 to January 2006. They collected data about users' online behavior relative to their geographical location and bandwidth consumption. Their results [GDJo6b] showed the diurnal and work-week online behavior of users, regardless of geographical location. The median online session time of supernodes was 5.5 hours. In spite of the fact that supernodes are relaying VoIP traffic, the average consumption of their bandwidth was relatively low. In [XY07] Xie and Yang investigated the impact of various factors on the VoIP quality, such as access capacities and AS policy constraints. Although these two studies contributed much to the understanding of Skype users' behavior, they did not give much insight into the geographical distribution of the peers in finer granularity. The number of the active users has grown more than double the size since 2006 and changes in the behavior of Skype users might have occurred. We ran a similar measurement study in order to capture that possible change and to get the time-based world map of the Skype users.

In the next sections we describe the process of collecting necessary data about the users' location and online times, as well as their analysis and implications on the geographical distribution and location-aware churn.

D.2 DATA COLLECTION

In order to understand the relations between geographical location and online user behavior, we need to collect the following data:

- geolocation of a user (derived from its IP address, using GeoIP databases such as [Max09]),

- time when the user joins the network, and
- online session time of the user.

Skype has a closed and proprietary protocol, a very limited API, and encrypts all TCP and UDP payloads. This makes the collection of the aforementioned data challenging. Possibilities for capturing the online times and the geolocation of ordinary nodes are very restricted. At the login phase, an ordinary peer contacts the Skype login server [BS06]. It communicates with its associated supernode(s) during the calls, searching the users, instant messaging and media exchange. Keep alive messages are exchanged every 120 seconds. This means that only the login server and a supernode can have the information about the arrival time and geolocation of a peer (through its IP address). Only a supernode knows when a peer went offline. It is, therefore, only possible to track a limited number of ordinary nodes, those peers that are associated to our supernode, to be used for measurements.

Capturing an hourly snapshot of online supernodes is possible by using hostcaches, like those presented in [GDJo6b, XY07]. A *hostcache* is a list of a supernodes contacts (IP address and port), which each node refreshes regularly. It is crucial to have at least one valid entry in the list. The IP addresses of the Skype supernodes can therefore be collected by analyzing the connections that a node establishes. A script we wrote for the data collection starts the client, which connects to an arbitrarily chosen supernode during the login procedure. A couple of seconds later, the script kills the client, which clears the hostcache. Upon restarting, the node gets a fresh hostcache and contacts a new supernode. The process is repeated, starting the client and killing it over and over again. The network activity was recorded using tcpdump [TCP08] running on the same node. By tracking connections that the node establishes, we can capture information about superpeers currently online. If the number of retrieved supernodes is close to the total number of all supernodes, it is possible to determine the online presence of each supernode over the time. The faster we gather supernodes, the more precise the captured uptimes are.

In [XY07], in each iteration, the hostcache is replaced with a list of just one supernode, randomly chosen from the previous list and marked as visited. The client is forced this way to get a fresh set of supernodes. Unfortunately, this method cannot be used in Skype binary versions greater than 1.4, as the hostcache is not stored in plain text format in the XML file anymore, but is encrypted. In order to capture as many supernodes as possible, our script was executed on six machines, each of them ran 40 to 50 unique clients with Skype 1.4 in parallel, in three different countries (Germany, Spain, and Canada). This way we examined the location-awareness of Skype network, checking whether or not an ordinary node gets the contacts of supernodes geographically close to it. All machines were behind a firewall to make sure none of the clients became a supernode. Tests showed that a client needs, on average 8 seconds to finish the entire joining process correctly. By running twenty clients in parallel on one virtual machine, we were

able to capture around 3,000 unique supernodes in one hour. In [XY07] 15,000 supernodes have been discovered in two hours and the number of active Skype users was 16.6 million at that time (July 2006). We concluded that we needed minimum 10×20 clients to capture all supernodes, as the number of the active Skype users doubled (32.0 millions in July 2008).

D.3 GEOGRAPHICAL DISTRIBUTION OF THE SUPERNODES

We ran experiments from the 12th of July to 20th July. Altogether more than 2.7 million unique IP addresses from 224 different countries have been observed and 5.9 GB of traffic has been captured. A map of all captured supernodes is presented in Figure 92.

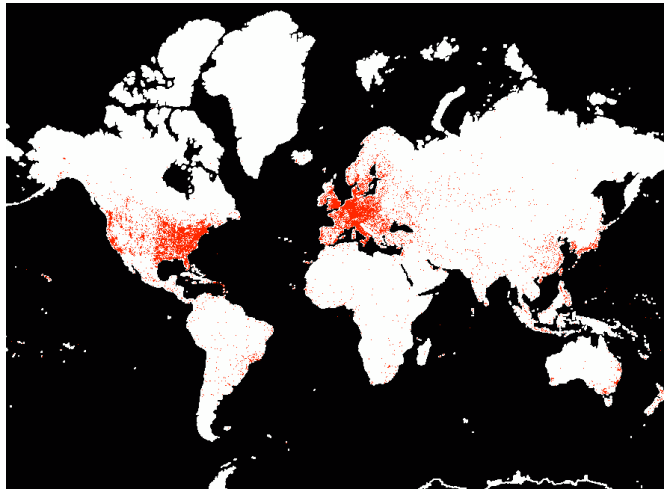


Figure 92: A map of all captured supernodes

We can see that the highest density of captured supernodes is in Europe and North America. Tables 12 and 13 show the number of supernodes per continent and in top ten countries populated by supernodes.

We can see that almost a half of the supernodes is in Europe. Supernodes in Asia contribute to 25% to the overall number, while 17% of supernodes are in North America. From European countries, Germany, UK, Italy and France have the biggest share of supernodes, while Japan, Taiwan and China (on surprising 10th place) are the leading Asian countries in number of Skype supernodes.

We relate these numbers to the number of Internet users per continent/country, presented in both tables (Table 12 and 13). Despite leading in the number of Internet users, Asia has a surprisingly small portion of Skype supernodes. The explanation could be that the Skype nodes in Asia are mostly publicly unreachable (behind firewall and NAT). Comparing these results to the statistics about all Skype users online [CK10], we can see that the number of active users follow the same distribution per continent like for the captured supernodes. That means

Continent	Supernodes	Percent	Internet Users	Percent
Europe	1 333 520	49.39 %	247 mio	18.29 %
Asia	674 015	24.96 %	656.3 mio	48.61 %
North America	458 927	17.00 %	286.3 mio	21.21 %
Oceania	125 752	4.66 %	15.09 mio	1.12 %
South America	76 045	2.82 %	94.41 mio	6.99 %
Africa	30 841	1.14 %	51.02 mio	3.78 %
unknown	1 042	0.04 %	-	-
Sum	2 700 142	100 %	1 350 120 000	100 %

Table 12: Number of captured supernodes per continent

Country	Supernodes	Percent	Internet Users	Percent
Germany	398 288	14.75 %	42.5 mio	3.15 %
USA	350 129	12.97 %	223 mio	16.52 %
Japan	187 560	6.95 %	88.11 mio	6.53 %
UK	178 778	6.62 %	40.2 mio	2.98 %
Taiwan	124 522	4.61 %	14.76 mio	1.09 %
Australia	104 326	3.86 %	4.277 mio	0.32 %
Canada	89 307	3.31 %	28 mio	2.07 %
Italy	89 036	3.30 %	32 mio	2.37 %
France	87 182	3.23 %	31.295 mio	2.32 %
China	82 281	3.05 %	253 mio	18.74 %
Sum	1 691 409	62.64 %	757.142 mio	56.09 %

Table 13: Top ten countries contributing to captured supernodes

that there are not only a small number of supernodes active in China, but all Skype users in general. Therefore, the argument about publicly unreachable Skype users in Asia is not valid. Another explanation can be the fact that most of Asian Internet users are coming from China (one third), where VoIP was illegal for long time [Ando6]. If we take into account that the majority of Kad users are exactly from China [SENBo7], this explanation could justify this phenomena.

Most of the captured supernodes are from Germany. As four out of six machines for our experiments were in Germany, we can assume that proximity plays a role when deciding which supernode from a hostcache to connect to. However, comparing our results to the results in [GDJo6b], measurements performed in USA, we can see clear similarities. Figures 93 and 94 show that the geographical distribution of the supernodes per continent is roughly the same in both measurements. In [GDJo6b], 45-60% of supernodes reside in Europe, 15-25% of

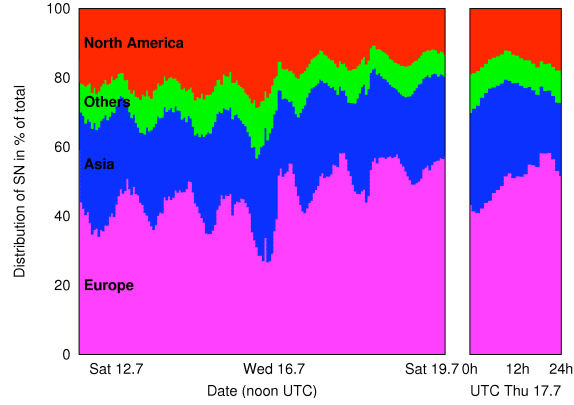


Figure 93: Geographic distribution of supernodes in our experiments

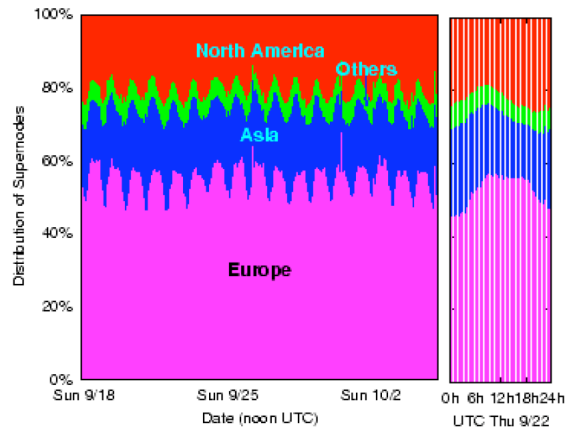


Figure 94: Geographic distribution of supernodes from [GD]06b]

supernodes are in North America and 20-25% are in Asia. We conclude, therefore, that the geographical proximity does not play an important role in supernode selection.

D.4 DIURNAL EFFECT OF SUPERNODE ONLINE BEHAVIOR

Figures 93 and 94 confirm the diurnal behavior of the number of online supernodes. On the right side of both graphics, the share of online supernodes per continent is magnified on the level of a day, July 17th 2008 and September 22nd 2005. We can see that most of the users are online during the working hours and less at night. For example, in Europe, the number of superpeers has a peak at 18h (UTC/GMT+1) and a minimum of supernodes online is at 6h in the morning. A similar situation is in North America and Asia. That means that independently of the location, diurnal behavior of online peers (or at least supernodes), based on their local time, remains the same. Figure 95 shows the comparison between two timezone groups with 14 hours difference (North America UTC-6, Far East UTC+8). We can see that the trend

of supernode activity in working hours, with a peak in the afternoon, holds for both.

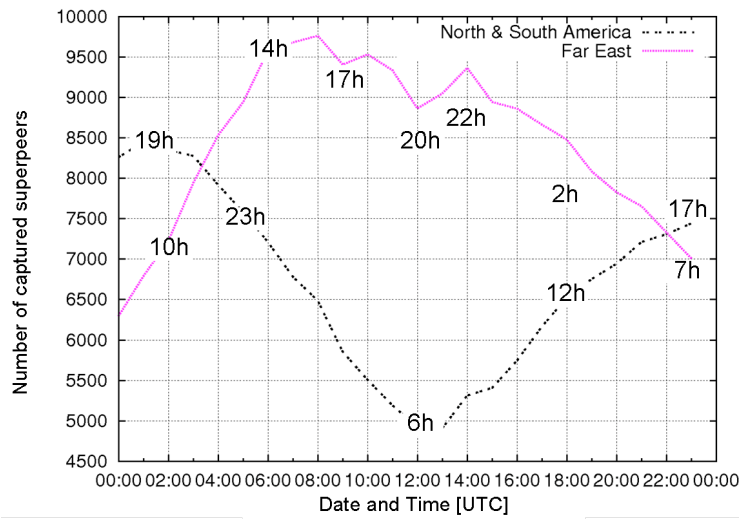


Figure 95: Number of the captured superpeers during a day in Asia and USA

Table 14 summarizes the basic statistical information of our measurements: maximum, minimum, mean and the standard deviation σ . The minimum population of supernodes is between 40 % to 50 % of the maximum population for the three biggest continents Asia, Europe and North America.

	Maximum	Minimum	Mean	Std.dev.
Africa	345	134	242.76	48.68
Asia	8 043	3 895	5 431.76	917.12
Europe	15 921	4 966	1499	1 202
North America	6 877	2 003	521	205
Oceania	11 473.05	3 339.37	955.97	618.51
South America	1 471.95	654.12	250.46	278.22

Table 14: Basic statistical values of supernodes' distribution by continents

For Germany, a maximum of 3 441 online supernodes was observed on Thursday, July 18th 2008 at 18h and a minimum of 1 825 online supernodes on the same day at 9h. The minimum number in this case is more than half (53.04 %) of the maximum number of online users.

In order to use this information about diurnal behavior of supernode-activities for a location-aware churn model, we have to take a closer look into the distribution of supernode session times.

D.5 SESSION TIMES OF SUPERNODES

The analysis of the supernode session times showed that more than 50% of the supernodes stayed online for a minimum of three hours, confirming the stability of the supernodes. A very small amount of peers were constantly online during the whole week (e.g. 18 in Germany).

Figure 96 show the number of supernodes staying online for more than x hours, for the top five countries. The results are presented in logarithmic scales in both x and y axis. We can recognize a very similar distribution for all five countries. In both graphs it is visible that the session times of the supernodes follow a power-law relationship regardless of their location.

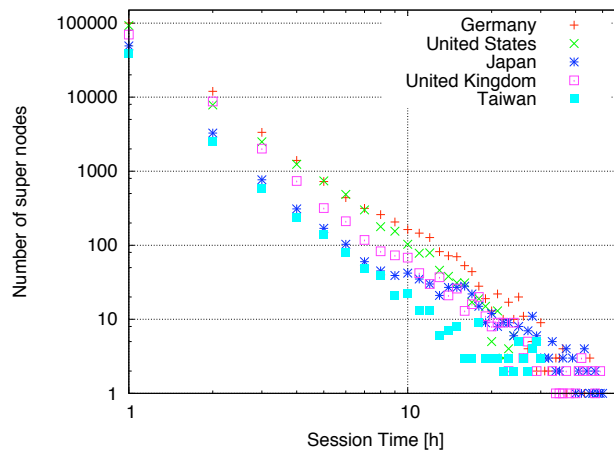


Figure 96: Session time of the supernodes in top 5 countries in logarithmical scale on both axis.

Figure 97 depicts the numbers of arriving and departing supernodes worldwide, from one hour to the next. The actual arrival and departure rates are nearly equal (16 000 to 20 000), so that the overall change is relatively small. Surprisingly, there is a strong correlation between the peer arrival and departure rates. The number of supernodes remaining online from one hour to the following is very stable at 5 000 supernodes. It is, actually more stable than the overall supernode population, which confirms the fact that supernodes with long session time are less likely to depart.

The maximum decrease in the number of online supernodes in the top 5 countries, from one hour to the following, tends to be at 13 % based on the earlier hour (see Table 15 for an overview of all countries). This confirms that the overall supernode population is fairly stable. This represents a beneficial user behavior for Globase.KOM, which exploits the advantage of longer available peers in the overlay structure.

To recall, we had two goals for our experiments on the Skype network, bearing in mind the related measurements studies [GDJ06b] and [XY07]. One goal was to obtain the geographical distribution of the peers in fine

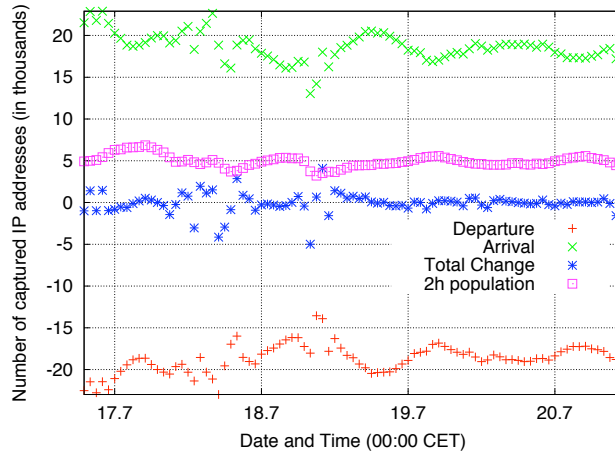


Figure 97: Number of supernodes arriving and departing the Skype network.

	Germany	UK	USA	Taiwan	Japan
Maximum drop	-12,46%	-37,92%	-13,09%	-15,95	-11,91%

Table 15: Relative maximum decrease of the total number of online supernodes from one hour to the following

granularity. The second goal was to examine the possible changes in the behavior and distributions of the Skype users, which a large growth in number of Skype users brought. Our measurements captured online supernodes, but not peers. Information about supernode behavior is valuable for our further evaluation as the overlay structure and routing of Globase.KOM relies directly on superpeers. The distribution of the Skype users on the world map can be scaled up using the captured distribution of supernodes.

Our measurement results showed that the distribution of the Skype supernodes over the world is somewhat different from the distribution of Internet users. Despite of the fact that almost half of the Internet users resides in Asia, just 25% of the Skype users are from Asia. The biggest ratio of Skype supernodes are from Europe. We observed strong diurnal behavior of supernodes, with the peaks in late afternoon and minima over the night, independently of the location, which did not signify any change from the measurements conducted in [GDJo6b]. We also concluded that the geographical proximity does not play an important role in supernode selection.

D.6 IMPLICATIONS ON A CHURN MODEL

Churn model describes the users' participation in a network using mathematical distribution functions. As our focus is a location-aware churn model, and providing a world map of peer density is an essential part of the model.

The fact that everywhere in the world supernodes behave diurnally makes the location-aware churn model easier. Churn model is certainly group based - each group includes peers in the similar timezones (three-hours zones). Each group is further separated into Benefactors, Peers, and Peepers (see Section 5.3.2), based on the length of their mean session time. The session time distribution is modeled as power law. A peer arrival can be modeled as a Poisson process with higher rates in working hours and lower in evenings, as proposed in [GDJ06b].

That would mean that the experiments should last a minimum of 24 hours when using such a model. We approximated the model by capturing the most critical time periods when the larger groups (e.g. Europe) have reached their minimum in size. For example, modeling the maximum drop of supernodes within an hour. For the resulting models, we need 2, 4 and 6 hours long experiments. Additionally, we performed experiments both on world- and country-level granularity and accordingly adapted the churn model.

A SIMULATION MODEL OF GLOBASE.KOM

A simulation model of Globase.KOM belongs to the overlay layer of the simulator. Figure 98 shows an implementation of Globase.KOM in the simulation framework PeerfactSim.KOM. As we can see, the GlobaseOverlayNode is batch generated by ComponentFactory of Globase.KOM, together with the pseudo-functionality layer factories in PeerfactSim.KOM, it consists of the following components:

- GlobaseCoordinates: The geographical position of peers, defined by two double values,
- GlobaseOverlayContact: Similar to IP addresses and ports in the real Internet world,
- GlobaseOverlayID: The simple overlay network identification, defined by an integer number,
- GlobaseZone: Only available in superpeer, a fixed area for which this superpeer is responsible,
- GlobaseOverlayRoutingTable contains different types of GlobaseOverlayContact i.e. parent superpeer, root superpeer, children superpeers and children peers,
- GlobaseOverlayRoutingInterconnection contains the interconnection with keywords HIGH, LOW, WIDE and NEAR (see Section 4.3.4).

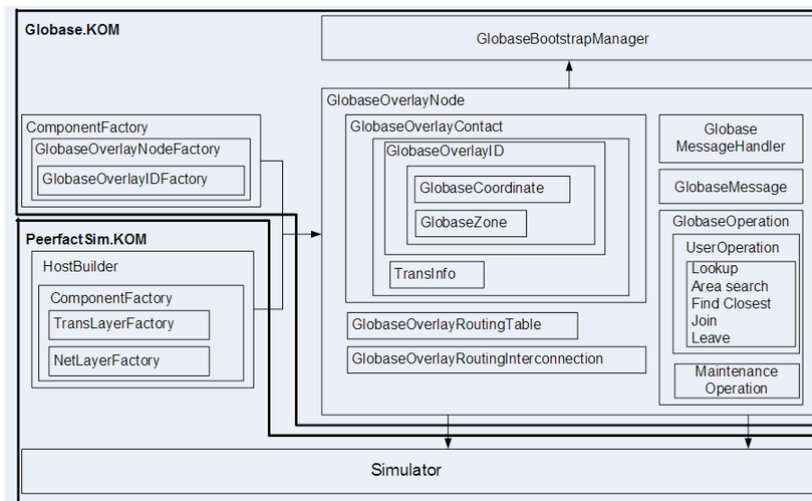


Figure 98: Simulation Model of Globase.KOM

All types of peers are supported (root, rootchild, superpeer, super-peer candidate, peer). The following types of peer are available: The

`GlobaseMessageHandler` is the central class which processes and dispatches the incoming messages. In `GlobaseOperation` some user specific functions are available such as: *join*, *lookup*, *search*, *find closest* and *leave*. It also implements periodic maintenance tasks.

The component `GlobaseCoordinates` not only stores the two coordinates for a peer in `Globase.KOM`, but also provides the following functionalities used within query resolution and maintenance operations:

- testing whether two peers have the same position
- calculating the distance between two pairs of coordinates
- testing whether a peer is located inside a give search area

The component `GlobaseZone` contains the position and dimension of the zone of responsibility for superpeers. It also provides the following functionalities used within query resolution and maintenance operations:

- calculating the size of a zone
- testing whether the zone is a ‘real’ zone (zones with edge lengths of 0 exist for non-superpeers)
- calculating the ratio between the long and the short edge of a zone
- calculating the distance from any given coordinates to the closest border of the zone
- testing whether any given coordinates lie within a zone
- testing whether a zone intersects with a given search area
- testing whether a zone contains a given search area
- testing whether a zone intersects with any given other zone
- testing whether a zone is the zone of the root superpeer
- creating a new, intersection-free zone within an existing zone

The clustering algorithm for creating new zones is described in detail in Section 4.5. Search areas are described by the coordinates of their center and the radius of the search area.

To implement features that are common for the different message types in `Globase.KOM` (e.g. hop count, sequence number) an abstract class `GlobaseMessage` was created. Over 20 different implementations of this abstract class were made to create the different message types of `Globase.KOM`. The different messages can be assigned to either user messages (e.g. `SEARCH`, `SEARCH_RESULT`, `LOOKUP_RESULT`, `NEW_PEER`, `PEER_OK`) or maintenance messages (e.g. `PING`, `KEEP_ALIVE`, `BECOME_SUPERPEER`, `NEW_ROOT`)

The component factory of `Globase.KOM` is `GlobaseOverlayNodeFactory`. For further details about the supplemented parameters and implementation, see Appendix F.

Visualizing Globase.KOM

In order to observe the shape of the zones, network traffic and specific critical situations (e.g. simultaneous failures of superpeers in the same branch) more easily, we developed a visualization of the `Globase.KOM` simulations. The visualization is a replay of simulations already run and therefore its speed can be adjusted and it can be paused or rewound. Visualization shows various metrics, such as exchanged messages per second or number of connections to each peer. Figure 99 shows a screenshot of the `Globase.KOM` simulation visualization.

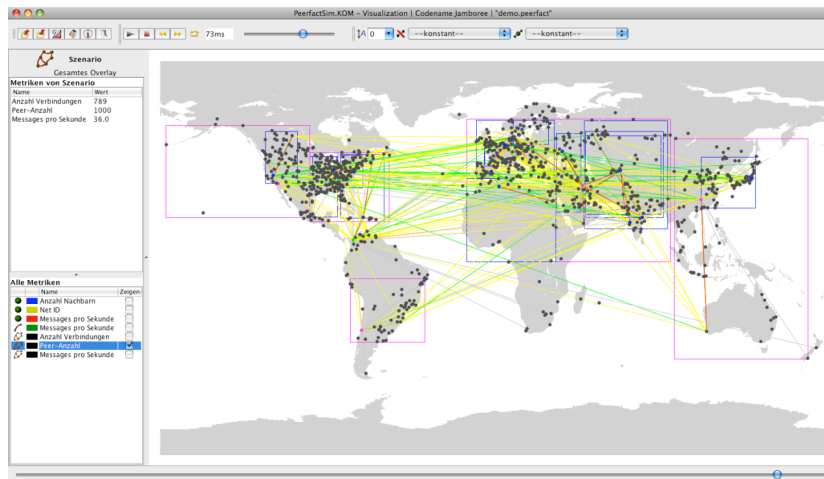


Figure 99: Visualization of the Globase.KOM while resolving user queries

To display messages, zones and different peer types, the visualization class `GlobaseAnalyzer` is used. When a message is sent or received, a method is called within the analyzer, which determines what has to be visualized. For example, if a `BECOME_SUPERPEER` message has been sent within the network, an edge will be added to the visualization (for a predefined time), the color of the receiving peer is changed and a rectangle for the new zone will be drawn. Every type of peer and message has an appropriate color (e.g. the root superpeer is green, superpeer and zone blue and regular peers black). The different message types in Globase.KOM are colored as green (lookup), yellow (search), red (find closest), blue (messages from the joining phase), and grey (all maintenance messages).

CONFIGURATION OF PEERFACTSIM.KOM

PeerfactSim.KOM is configured through a XML configuration file, in which required components for the simulation are chosen and set up according to the respective requirements. Specific information on the configuration of Globase.KOM are the subject of this appendix.

The network layer that has to be used for simulating Globase.KOM is GnpNetLayer. The GnpNetLayer offers a bitmap based distribution of peers using GnpBitmapNetLayerFactory, which allows the simulation of networks according to host positions derived from a grey-scale bitmap. The GnpBitmapNetLayerFactory can be found in the package `de.tud.kom.p2psim.impl.network.gnp` of PeerfactSim.KOM. As for the latency model a modified version of the GnpLatencyModel was used. Minimum and maximum latency for this model can be set using the parameters `minLatency` and `maxLatency`. As bandwidth manager, the GnpNetBandwidthManagerPeriodical from the simulator package mentioned above, was used.

As the transport layer and host builder for the simulations, the base implementations DefaultTransLayerFactory and DefaultHostBuilder were used.

To simulate Globase.KOM, the component factory used by PeerfactSim.KOM must be set to GlobaseOverlayNodeFactory from package `de.tud.kom.p2psim.impl.overlay.globase`. The protocol parameters that need to be set for this class are explained in Table 16.

<i>Protocol parameter</i>	<i>Description</i>
L_1	Threshold for load level "Overloaded"
L_2	Threshold for load level "Critically Overloaded"
P_1	Maintenance periods for superpeers
P_2	Maintenance periods for child peers
P_3	Maintenance periods of brothers to root children
P_4	Period for load balancing task
R_1	Hotspot ratio
S_1	Size of interconnections
S_2	Size of cache
T_1	Timeout for <i>Find Closest</i> operation
T_2	Timeout for <i>Search</i> operation

Table 16: Protocol parameters for Globase.KOM

The scenarios used in the simulations of Globase.KOM were stored in Comma Separated Values (CSV) files. To simulate scenarios from these CSV files, the CSVScenarioFactory from the package `de.tud.kom.p2psim.impl.scenario` of PeerfactSim.KOM was used. The component class for the scenario needs to be set to GlobaseOverlayNode, which can be found in the Globase.KOM package mentioned above.

A sample configuration file for simulating Globase.KOM can be found

```
<?xml version='1.0' encoding='utf-8'?>
<Configuration>
  <Default>
    <Variable name="seed" value="36971" />
    <Variable name="finishTime" value="22m" />
    <Variable name="actions" value="config/actions.dat"
      />
    <Variable name="gnpDataFile" value="data/
      measured_data.xml"/>
    <Variable name="size" value="10000"/>
  </Default>
  <SimulatorCore class="de.tud.kom.p2psim.impl.simengine.
    Simulator"
    static="getInstance" seed="$seed" finishAt="$
      finishTime" />

  <NetLayer
    class="de.tud.kom.p2psim.impl.network.gnp.
      GnpBitmapNetLayerFactory"
    downBandwidth="500" upBandwidth="250" PbaPeriod="1"
    experimentSize="$size" bitmapPath="config/worldmap.
      bmp">
    <LatencyModel
      class="de.tud.kom.p2psim.impl.network.gnp.
        GnpLatencyModel"
      minLatency="50" maxLatency="250" />
    <BandwidthManager
      class="de.tud.kom.p2psim.impl.network.gnp.
        GnpNetBandwidthManagerPeriodical"/>
  </NetLayer>

  <TransLayer
    class="de.tud.kom.p2psim.impl.transport.
      DefaultTransLayerFactory"/>

  <ComponentFactory
    class="de.tud.kom.p2psim.impl.overlay.globase.
      GlobaseOverlayNodeFactory"
    L1="400" L2="800" P1="60" P2="120" P3="60" P4="10"
    R1="0.3" S1="20" S2="10" T1="2" T2="2"/>

  <Monitor class="de.tud.kom.p2psim.impl.common.
    DefaultMonitor"
    start="on" stop="$finishTime">
    <Analyzer class="de.tud.kom.p2psim.impl.overlay.
      globase.analyzers.GlobaseAvgHopsAnalyzer"/>
    <Analyzer class="de.tud.kom.p2psim.impl.overlay.
      globase.analyzers.GlobaseAvgODAnalyzer"/>
    <Analyzer class="de.tud.kom.p2psim.impl.overlay.
      globase.analyzers.GlobaseAvgRDPAnalyzer"/>
  </Monitor>
</Configuration>
```

```

        <Analyzer class="de.tud.kom.p2psim.impl.overlay.
            globase.analyzers.
            GlobaseLoadBalanceRatioAnalyzer"/>
        <Analyzer class="de.tud.kom.p2psim.impl.util.vis.
            adapter.analyzers.GlobaseAnalyzer"/>
    </Monitor>

    <HostBuilder
        class="de.tud.kom.p2psim.impl.scenario.
            DefaultHostBuilder"
        experimentSize="$size">

        <Group groupID="g1" size="1000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

        <Group groupID="g2" size="2000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

        <Group groupID="g3" size="1000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

        <Group groupID="g4" size="1000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

        <Group groupID="g5" size="5000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

    </HostBuilder>

    <Scenario class="de.tud.kom.p2psim.impl.scenario.
        CSVScenarioFactory"
        actionsFile="$actions"
        componentClass="de.tud.kom.p2psim.impl.overlay.
            globase.GlobaseOverlayNode" >
    </Scenario>

</Configuration>

```

Example configuration file for the simulation of Globase.KOM:

```

<?xml version='1.0' encoding='utf-8'?>
<Configuration>
  <Default>
    <Variable name="seed" value="36971" />
    <Variable name="finishTime" value="2m" />
    <Variable name="actions" value="config/actions.dat" />
    <Variable name="gnpDataFile" value="data/measured_data.xml"/>
    <Variable name="size" value="10000"/>
  </Default>
  <SimulatorCore class="de.tud.kom.p2psim.impl.simengine.Simulator"
    static="getInstance" seed="$seed" finishAt="$finishTime" />

  <NetLayer
    class="de.tud.kom.p2psim.impl.network.gnp.GnpBitmapNetLayerFactory"
    downBandwidth="500" upBandwidth="250" PbaPeriod="1"
    experimentSize="$size" bitmapPath="config/worldmap.bmp">
    <LatencyModel
      class="de.tud.kom.p2psim.impl.network.gnp.GnpLatencyModel"
      minLatency="50" maxLatency="250" />
    <BandwidthManager
      class="de.tud.kom.p2psim.impl.network.gnp.GnpNetBandwidthManagerPeriodical"/>
  </NetLayer>

  <TransLayer
    class="de.tud.kom.p2psim.impl.transport.DefaultTransLayerFactory"/>

  <ComponentFactory
    class="de.tud.kom.p2psim.impl.overlay.globase.GlobaseOverlayNodeFactory"
    L1="400" L2="800" P1="60" P2="120" P3="60" P4="10"
    R1="0.3" S1="20" S2="10" T1="2" T2="2"/>

  <Monitor class="de.tud.kom.p2psim.impl.common.DefaultMonitor"
    start="on" stop="$finishTime">
    <Analyzer class="de.tud.kom.p2psim.impl.overlay.globase.analyzers.GlobaseAvgHopsAnalyzer"/>
    <Analyzer class="de.tud.kom.p2psim.impl.overlay.globase.analyzers.GlobaseAvgODAnalyzer"/>
    <Analyzer class="de.tud.kom.p2psim.impl.overlay.globase.analyzers.GlobaseAvgRDPAnalyzer"/>
    <Analyzer class="de.tud.kom.p2psim.impl.overlay.globase.analyzers.GlobaseLoadBalanceRatioAnalyzer"/>
  </Monitor>
</Configuration>

```



```

        <Analyzer class="de.tud.kom.p2psim.impl.util.vis.
            adapter.analyzers.GlobaseAnalyzer"/>
    </Monitor>

    <HostBuilder
        class="de.tud.kom.p2psim.impl.scenario.
            DefaultHostBuilder"
        experimentSize="$size">

        <Group groupID="g1" size="1000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

        <Group groupID="g2" size="2000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

        <Group groupID="g3" size="1000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

        <Group groupID="g4" size="1000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

        <Group groupID="g5" size="5000">
            <NetLayer />
            <TransLayer />
            <ComponentFactory />
        </Group>

    </HostBuilder>

    <Scenario class="de.tud.kom.p2psim.impl.scenario.
        CSVScenarioFactory"
        actionsFile="$actions"
        componentClass="de.tud.kom.p2psim.impl.overlay.
            globase.GlobaseOverlayNode" >
    </Scenario>

</Configuration>

```


SUPPLEMENTARY DETAILS TO CHAPTER 6: PARAMETER SETTINGS

This Appendix provides the detailed simulation and experiment settings for the evaluation results presented in the Section 6.

For the setting protocol parameters L_1 , L_2 , the number of interconnections, size of cache, timeouts of queries, and parameters for failure detection mechanisms, the following simulation settings are taken:

<i>Protocol parameters</i>	<i>Value(s)</i>
Maintenance periods for superpeers P_1 , P_3	60 sec.
Maintenance period for reg. peers P_2	120 sec.
Load balancing period P_4	10 sec.
Hotspot ratio R_1	0,3
Size of cache S_2	10
Timeout for operations T_1 , T_2	2 sec.
<i>Simulator settings</i>	<i>Value(s)</i>
Number of peers	1 000, 5 000, 10 000
Experiment Timeline	Efficiency Timeline

Table 17: Parameters and settings for the parameter calibration
Parameters for the examination of the effects of L_2 :

<i>Protocol parameters</i>	<i>Value(s)</i>
Ratio $\nu = \frac{L_1}{L_2}$	0.5
Load threshold L_2	10, 20, 30, 40, 50, 60, 70, 80, 90, 100
Number of interconnections S_1	20
<i>Simulator settings</i>	<i>Value(s)</i>
Number of peers	1 000, 5 000, 10 000
Churn	None

Table 18: Parameters and settings for the evaluation of load threshold L_2 (detailed simulation settings are given in Appendix F)

<i>Protocol parameters</i>	<i>Value(s)</i>
Ratio $\nu = \frac{L_1}{L_2}$	0.3, 0.4, 0.5, 0.6
Load threshold L_2	60, 70, 80
Number of interconnections S_1	20
<i>Simulator settings</i>	<i>Value(s)</i>
Number of peers	1 000, 5 000, 10 000
Churn	None

Table 19: Parameters and settings for the effects of ratio $\nu = \frac{L_2}{L_1}$

<i>Protocol parameters</i>	<i>Value(s)</i>
Load threshold L_1	50
Load threshold L_2	100
Number of interconnections S_1	0, 10, 20, ∞
<i>Simulator settings</i>	<i>Value(s)</i>
Number of peers	10 000
Churn	adapted Skype-churn model

Table 20: Parameters and settings for the evaluation of interconnections

<i>Protocol parameters</i>	<i>Value(s)</i>
Load threshold L_1	50
Load threshold L_2	100
Maintenance periods for superpeers P_1, P_3	60 sec.
Maintenance period for reg. peers P_2	120 sec.
Load balancing period P_4	10 sec.
Hotspot ratio R_1	0,3
Number of interconnections S_1	20
Size of cache S_2	10
Timeout for operations T_1, T_2	2 sec.

Table 21: Setting of parameters for the experiments

LIST OF THE MESSAGE TYPES IN GLOBASE.KOM PROTOTYPE

Exchanged messages in the Globase.KOM overlay protocol can be separated into three main categories

- *Notifications* - these are messages that are used to inform the receiving party of a specific event and do not require a reply,
- *Requests* - these messages are used to request some information from the receiving party and a reply is expected,
- *Responses* - these are messages sent back as a reply to a previously received request.

The following is the full list used in all messages of the Globase.KOM prototype:

NOTIFICATIONS

- *BecomeSuperPeerMsg* - notifies the receiving peer that it is elected to become a superpeer with the zone of responsibility specified in the message.
- *NewParentMsg* - notifies the receiving peer of its new superpeer.
- *NewBrotherMsg* - notifies peers about the arrival of a new brother peer.
- *BrotherLostMsg* - notifies peers about a dead or offline brother peer.
- *ExceptionMessage* - notifies the receiving peer about the occurred exception, including detailed information.
- *KeepAliveMsg* - messages periodically exchanged between the peers to indicate that they are still online and responding.
- *LeaveMsg* - announces the departure of a peer.

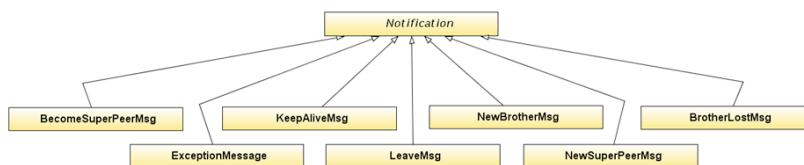


Figure 100: Notifications in the Globase.KOM prototype

REQUESTS

- *JoinRequest* is being sent when a peer joins the network. It contains the peer's coordinates and meta-data.
- *RejoinRequest* is a request issued by peers that already have their IDs but were temporally disconnected or lost connection to their superpeers.
- *LookupRequest* is classified as a user generated message. It is used to request information about a peer, based on its coordinates.

- *FindClosestRequest* is a user generated message. It requests information about the peer that is geographically closest to the specified coordinates.
- *SearchRequest* is also user generated. It is used to perform searches for peers positioned within an area of a given radius.

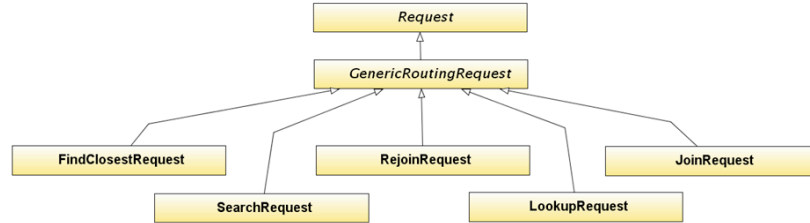


Figure 101: Requests in the Globase.KOM prototype

GenericRoutingRequest extensions (Figure 101) are requests that contain routing coordinates and need to be routed to the specified destination. Such requests are handled by the GenericRoutingHandler (see Section 7.2.4).

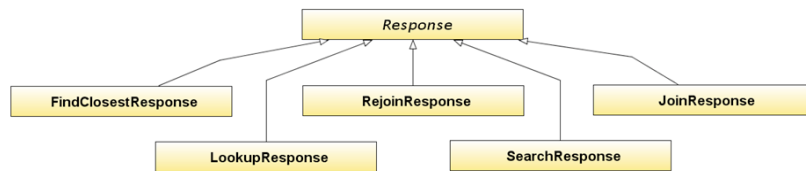


Figure 102: Responses in the Globase.KOM prototype

RESPONSES

- *JoinResponse* is the answer to a join request. It contains the ID generated for the requesting peer and other important information (superpeer, root, neighbors, etc.)
- *RejoinResponse* is the answer to a rejoin request. It contains information regarding the new superpeer of the rejoining node and other specific data.
- *LookupResponse* is the answer to a lookup request. It contains information about the peer positioned at the coordinates specified in the lookup request.
- *FindClosestResponse* is the answer to a find closest request. It contains the above mentioned information.
- *SearchResponse* is the response to a search request containing all peers positioned in the area specified in the search request.

Messages that change the network structure, such as *BecomeSuperPeerMsg*, *NewParentMsg*, etc. are only accepted if they are issued by the parent of the receiving peer. Currently, we only compare the claimed initiator's ID and the parent's ID. Although the faking of IDs is not avoided and detected, the danger of applying changes after having mistakenly received or sent a message of these types (e.g. because of an error or miscalculation) is significantly reduced.

LIST OF FIGURES

Figure 1	Example binary space partitioning	12
Figure 2	KD-Tree	12
Figure 3	R-Tree	13
Figure 4	Four iterations of Hilbert and z- space filling curves	13
Figure 5	Area search using Prefix Hash Tables (Source: [CRR ⁺ 05])	14
Figure 6	Example of 4-level Skip List with the routing path to the key 13	16
Figure 7	Illustration of the RectNet overlay structure	17
Figure 8	Example of an area search. Node n_1 is the initiator of the search.	18
Figure 9	Centralized peer-to-peer overlay network	23
Figure 10	Structured pure peer-to-peer overlay network	23
Figure 11	Unstructured pure peer-to-peer overlay network	24
Figure 12	Hybrid peer-to-peer overlay network	25
Figure 13	Hierachical peer-to-peer overlay network	25
Figure 14	Gia topology adaptation: low-capacity peers keep connections to high-capacity peers and use them for routing	31
Figure 15	Example of the query resolution in Chord	32
Figure 16	Example of the query resolution in Kademlia	32
Figure 17	Gnutella 0.6 ultrapeers and leaves	33
Figure 18	Maintenance strategies	35
Figure 19	Stabilization process upon a peer join (fixing a broken ring)	36
Figure 20	Hit rate with the increasing network size	41
Figure 21	The average query response time. Only succeeded queries are included	41
Figure 22	Share of stale messages	42
Figure 23	Average number of hops needed to resolve a query	42
Figure 24	Distribution of the consumed download bandwidth per peer	44
Figure 25	Decrease of query success when 50% of the peers simultaneously fail at t_0 (except Chord where only 10% peers fail)	46
Figure 26	Share of stale messages when 50% of the peers simultaneously fail at t_0 (except Chord where only 10% peers fail)	46
Figure 27	Basic structure of the Globase.KOM overlay	51
Figure 28	Structure of Peer Identifier Space	53
Figure 29	Management of the resource identifier space	55
Figure 30	Basic routing topology of the Globase.KOM	57
Figure 31	Interconnections	57
Figure 32	Example of an area search	60
Figure 33	Example of the <i>find the closest</i> query	61
Figure 34	Load levels and thresholds in Globase.KOM	65
Figure 35	Overlapping zones during clustering	68
Figure 36	Overlapping of zones resolved	69
Figure 37	Efficiency experiment timeline	78
Figure 38	Stability experiment timeline	79

Figure 39	Robustness experiment timeline	79
Figure 40	The lifetime of a peer consisting of several session intervals [PKL ⁺ 08]	80
Figure 41	Measuring recovery and variation for stability and robustness	82
Figure 42	Functionality layers of PeerfactSim.KOM	84
Figure 43	Geographical maps colored with grayscale reflecting the concentration of users - the darker parts represent areas with more users relative to lighter parts on the map	85
Figure 44	Zones and corresponding basic superpeer tree for low $L_2 = 30$	89
Figure 45	Zones and corresponding basic superpeer tree for high $L_2 = 100$	89
Figure 46	Shape of superpeer tree for different L_2 and 1 000 peers	90
Figure 47	Effects of L_2 on query performance and costs	91
Figure 48	Shape of superpeer tree for different ratio $\nu = \frac{L_1}{L_2}$	92
Figure 49	Superpeer tree for low ratio $\frac{L_1}{L_2}$ ($L_2 = 70$ and $L_1 = 21$)	93
Figure 50	Superpeer tree for high ratio $\frac{L_1}{L_2}$ ($L_2 = 70$ and $L_1 = 42$)	93
Figure 51	Effects of ν ratio on query performance and costs	94
Figure 52	Effects of number of interconnection on number of hops for queries	95
Figure 53	Effects of number of interconnection on response time for queries	97
Figure 54	Effects of number of interconnection on costs	98
Figure 55	Query performance of Globase.KOM, Chord, and Kademlia with 10 000 peers	101
Figure 56	Distribution of incoming and outgoing traffic	101
Figure 57	Percentage of received messages	102
Figure 58	Variation of query response time and hop count when peers/superpeers simultaneously leave the network	104
Figure 59	Variation of retrievability when peers/superpeers simultaneously leave the network	105
Figure 60	Variation of relative delay penalty and load balance ratio time when peers/superpeers simultaneously leave the network	105
Figure 61	Variation of hop count, response time, and retrievability with more frequent user queries	106
Figure 62	Variation of relative delay penalty, load balance ratio and retrievability with more frequent user queries	107
Figure 63	Variation of hop count and response time with intensive peer failures	108
Figure 64	Variation of retrievability with intensive peer failures	109
Figure 65	Variation of relative delay penalty and load balance ratio with intensive peer failures	110
Figure 66	Variation of observed metrics when multiple superpeers from the same tree branch simultaneously fail	110
Figure 67	Overlay structure of RectNet (left) and corresponding responsibility zones (right)	111
Figure 68	Response times of Globase.KOM and RectNet	112

Figure 69	Comparison of Relative Delay Penalty	113
Figure 70	World map and search panel of our prototype user interface	119
Figure 71	Statistics panel of our prototype user interface	120
Figure 72	Package organization of the prototype	121
Figure 73	The three main functional layers and interfaces between them	121
Figure 74	Architecture of the network layer in the Globase.KOM prototype	122
Figure 75	Interface between overlay and application - globase.core.Application	123
Figure 76	The three generic message types in the Globase.KOM prototype	123
Figure 77	Handlers in the Globase.KOM prototype	124
Figure 78	Sample control flow at message submission/arrival	125
Figure 79	Collaboration between the main elements involved in the representation of interconnections	126
Figure 80	Events in the Globase.KOM prototype	127
Figure 81	Collaboration between LocalNode and the Daemons	128
Figure 82	Supported implementations of PositionLocator - the interface used to determine the coordinates of the single peers	128
Figure 83	Sample density world map	129
Figure 84	Resulting peer distribution	129
Figure 85	Bandwidth consumption and response time of a peer in Globase.KOM prototype	131
Figure 86	A set of 40 documents with their relative frequency, Zipf-distributed	165
Figure 87	Experiment Timeline	167
Figure 88	Representation of a point on the earth's surface in the Geographical Coordinate System (Source: [IBMo8])	169
Figure 89	Three types of map projections (Source: [MSNo9])	170
Figure 90	Plate Carée projection (Source: [Mat09])	170
Figure 91	Skype Network	174
Figure 92	A map of all captured supernodes	176
Figure 93	Geographic distribution of supernodes in our experiments	178
Figure 94	Geographic distribution of supernodes from [GDJo6b]	178
Figure 95	Number of the captured superpeers during a day in Asia and USA	179
Figure 96	Session time of the supernodes in top 5 countries in logarithmical scale on both axis.	180
Figure 97	Number of supernodes arriving and departing the Skype network.	181
Figure 98	Simulation Model of Globase.KOM	183
Figure 99	Visualization of the Globase.KOM while resolving user queries	185
Figure 100	Notifications in the Globase.KOM prototype	195
Figure 101	Requests in the Globase.KOM prototype	196

LIST OF TABLES

Table 1	Parameters for evaluation of effects of load threshold L_1 (a) and ratio ν (b) on the shape and size of the tree	88
Table 2	Parameter settings for the evaluation of efficiency	100
Table 3	Simulator settings for the evaluation of stability	104
Table 4	Simulator settings for the evaluation of robustness	107
Table 5	Simulation settings for the comparative evaluation of Globase.KOM and RectNet	112
Table 6	Timeline of experiment on prototype	130
Table 7	Churn Model Parameters	166
Table 8	Chord: parameter settings	167
Table 9	Kademlia: parameter settings	167
Table 10	Gia: parameter settings	168
Table 11	Gnutella 0.6: parameter settings	168
Table 12	Number of captured supernodes per continent	177
Table 13	Top ten countries contributing to captured supernodes	177
Table 14	Basic statistical values of supernodes' distribution by continents	179
Table 15	Relative maximum decrease of the total number of on-line supernodes from one hour to the following	181
Table 16	Protocol parameters for Globase.KOM	187
Table 17	Parameters and settings for the parameter calibration	193
Table 18	Parameters and settings for the evaluation of load threshold L_2 (detailed simulation settings are given in Appendix F)	193
Table 19	Parameters and settings for the effects of ratio $\nu = \frac{L_2}{L_1}$	194
Table 20	Parameters and settings for the evaluation of interconnections	194
Table 21	Setting of parameters for the experiments	194

ACRONYMS

1D	One-dimensional
2D	Two-dimensional
AS	Autonomous System
C/S	Client/Server
CAN	Content Addressable Network
CET	Central European Time, UTC + 1 hour
CPU	Central Processing Unit
DHT	Distributed Hash Table

DMS	Degrees Minutes Seconds
DNS	Domain Name System
GMT	Greenwich Mean Time
GPS	Global Positioning System
GSD	Global Software Development
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
ISP	Internet Service Provider
LBR	Load Balance Ratio
MBR	Minimum Bounding Rectangles
MMOGs	Massively Multiplayer Online Games
MSB	Most Significant Bit
NAT	Network Address Translation
ns-2	(Berkley) Network Simulator-2
P2P	Peer-to-Peer
PSTN	Public Switched Telephone Networks
QoS	Quality of Service
QRT	Query Response Time
RDP	Relative Delay Penalty
RTT	Round-Trip-Time
SN	Supernode
TTL	Time-to-Live
UID	Unique Identifier
UP	Ultrapeer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Coordinated Universal Time, equivalent for Greenwich Mean Time
VoIP	Voice-over-IP
WWW	World Wide Web

CURRICULUM VITÆ

PERSONAL DETAILS

Name: Aleksandra Kovačević

Date and Place of Birth: 21st June 1980 in Belgrade, Serbia

Nationality: Serbian

<http://www.aleksandrakovacevic.com>

EDUCATION

- | | |
|----------------------------|--|
| <i>June 2005 - present</i> | Doctoral candidate at the Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt.
Topic: <i>Peer-to-Peer Location-based Search: Engineering a novel Peer-to-Peer Overlay Network</i> |
| <i>1999 - 2004</i> | Studies of Computer Science and Computer Engineering, School of Electrical Engineering, University of Belgrade
Third graduated student in the whole generation (860 registered students)
Degree: Dipl. Ing., M.Sc. equivalent
Grade: 7,76/10 (10 being maximum) |
| <i>1995 - 1999</i> | First Belgrade High School (rated as the best general education high-school in Belgrade)
Major in exact sciences
Grade: 5,0/5 (5 being maximum) |

ACADEMIC EXPERIENCE

- | | |
|----------------------------|---|
| <i>June 2008 - present</i> | Head of the research group "Peer-to-Peer Networking" at Multimedia Communications Lab, Technische Universität Darmstadt |
| <i>June 2006 - present</i> | Researcher on DFG (German Research Foundation) Research Unit (DFG FG FOR 733) QuaP2P Project "Improving the Quality of Peer-to-Peer Systems by Systematically Researching Quality Features and their Interdependencies" |
| <i>June 2005 - present</i> | Researcher at the Multimedia Communications Lab, Department of Electrical Engineering and Information Technology, Technische Universität Darmstadt |
| <i>2004 - 2005</i> | Researcher at IPSI, Belgrade |

AWARDS

- | | |
|-------------|--|
| <i>1999</i> | Awarded as the best mathematician of the generation in high school (around 300 students)
Won 2nd prize at a district competition in mathematics |
|-------------|--|

Qualified for the state competition in mathematics
and placed among the best 10 competitors

SUPERVISED DIPLOMA, M.SC., STUDENT, AND B.SC. THESES

Andre Mink "P2P-Systeme fuer lokationsbasierte Dienste" (Diploma thesis)

Sebastian Kaune "Evaluation von strukturierten P2P-Systemen im Hintergrund von Katastropheneinsaetzen" (Diploma thesis)

Joerg Weckbach "Design und Implementierung einer P2P-Anwendung fuer lokationsbasierte Dienste" (Diploma thesis)

Till Kofink "Evaluation of a P2P Overlay for Location-based Search" (Diploma thesis)

Hans Heckel "Enhancement of a Large-Scale P2P Simulator" (Studienarbeit)

Inaki Maya "Taxonomy and Analytical Comparison of P2P Location-based Search Mechanism" (Bachelor thesis)

Katharina Mohr "Lokations-basierte Suche in P2P Overlay Netzwerken" (Bachelor thesis)

Christian Kleinboeltig "Capturing and Modelling the Behaviour of Skype Users" (Studienarbeit)

Aleksandar Todorov "Implementation and Visualization of a P2P System for Location-based Search" (Bachelor thesis)

Robert Schadler "Entwicklung und Simulation von Mechanismen in unstrukturierten P2P-Overlay-Netzwerken" (Bachelor thesis)

Pei Wang "Robustness und Stability of P2P Overlay Network for Location-based Search" (Diploma thesis)

Jonas Kuehne "Implementierung und Evaluation von unstrukturierten, hybriden und strukturierten P2P Netzwerken" (Diploma thesis)

Guy Kopf "Failure Detection Mechanisms in P2P Overlays" (Bachelor thesis)

Leonid Melnyk "Evaluation of Hierarchical P2P Overlays" (Bachelor thesis)

Leo Nobach "Assessing Quality of P2P Search Overlays" (Bachelor thesis)

Patrick Fongue "Load Balancing in P2P Overlays" (Bachelor thesis)

Nico Rottstädt "Development of P2P Application for Location-based Search" (Diploma thesis, running)

PUBLICATIONS

JOURNAL ARTICLES

- [1] Aleksandra Kovacevic, Oliver Heckmann, Nicolas Liebau, and Ralf Steinmetz. Location Awareness - Improving Distributed Multimedia Communication. *Special Issue of the Proceedings of IEEE on Advances in Distributed Multimedia Communications*, 96(1), January 2008.
- [2] Aleksandra Kovacevic, Sebastian Kaune, Patrick Mukherjee, Nicolas Liebau, and Ralf Steinmetz. Benchmarking Platform for Peer-to-Peer Systems. *it - Information Technology (Methods and Applications of Informatics and Information Technology)*, 49(5):312–319, September 2007.
- [3] Kalman Graffi, Aleksandra Kovacevic, Patrick Mukherjee, Michael Benz, Christof Leng, Dirk Bradler, Julian Schroeder-Bernhardi, and Nicolas Liebau. Peer-to-Peer Forschung - überblick und Herausforderungen. *it - Information Technology (Methods and Applications of Informatics and Information Technology)*, 49(5):272–279, September 2007. DOI 10.1524/itit.2007.49.5.279.

CONFERENCE AND WORKSHOP CONTRIBUTIONS

- [1] Kalman Graffi, Patrick Mukherjee, Burkhard Menges, Daniel Hartung, Aleksandra Kovacevic, and Ralf Steinmetz. Practical Security in P2P-based Social Networks. In *Proceedings of the 34th Annual IEEE Conference on Local Computer Networks (LCN)*, October 2009.
- [2] Osama Abboud, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Quality Adaptation Mechanisms for P2P Video Streaming Systems. In *To appear in the 12th IFIP/IEEE International Conference on Management of Multimedia and Mobile Networks and Services (MMNS 2009)*, October 2009.
- [3] Kalman Graffi, Dominik Stingl, Julius Rückert, and Aleksandra Kovacevic. Monitoring and Management of Structured Peer-to-Peer Systems. In *Proceedings of 9th International Conference on Peer-to-Peer Computing 2009*, September 2009.
- [4] Osama Abboud, Aleksandra Kovacevic, Kalman Graffi, Konstantin Pussep, and Ralf Steinmetz. Underlay Awareness in P2P Systems: Techniques and Challenges. In *Proceedings IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, May 2009.
- [5] Aleksandra Kovacevic, Aleksandar Todorov, Nicolas Liebau, Dirk Bradler, and Ralf Steinmetz. Demonstration of a Peer-to-Peer Approach for Spatial Queries. In *Proceedings of Database Systems for Advanced Applications (DASFAA'09)*, April 2009.

- [6] André König, Aleksandra Kovacevic, and Ralf Steinmetz. Issues, Challenges and Opportunities of Setting up Experimental Peer-to-Peer Facilities - A Personal Story. In Udo Bub, Anastasius Gavras, Thomas Magedanz, and Phuoc Tran-Gia, editors, *Panlab Workshop on Setup and Operation of Open Testbed Infrastructures in the Context of NGN and Future Internet - Status Quo and Quo Vadis in conjunction with 16. ITG/GI Fachtagung Kommunikation in Verteilten Systemen (KiVS) 2009*, March 2009.
- [7] Sebastian Kaune, Konstantin Pussep, Christof Leng, Aleksandra Kovacevic, Gareth Tyson, and Ralf Steinmetz. Modelling the Internet Delay Space Based on Geographical Locations. In *Proceedings of the 17th Euro-micro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pages 301–310, February 2009.
- [8] Aleksandra Kovacevic, Kalman Graffi, Sebastian Kaune, Christof Leng, and Ralf Steinmetz. Towards Benchmarking of Structured Peer-to-Peer Overlays for Network Virtual Environments. In *Proceedings of the 14th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 799–804, December 2008.
- [9] Kalman Graffi, Aleksandra Kovacevic, Song Xiao, and Ralf Steinmetz. SkyEye.KOM: An Information Management Over-Overlay for Getting the Oracle View on Structured P2P Systems. In *Proceedings of the 14th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, pages 279–286, December 2008.
- [10] Kalman Graffi, Sergey Podrajanski, Patrick Mukherjee, Aleksandra Kovacevic, and Ralf Steinmetz. A Distributed Platform for Multimedia Communities. In *Proceedings of IEEE International Symposium on Multimedia (ISM)*, page 6, December 2008.
- [11] Sebastian Kaune, Tobias Lauinger, Aleksandra Kovacevic, and Konstantin Pussep. Embracing the Peer Next Door: Proximity in Kademlia. In *Proceedings of the 8th International Conference on Peer-to-Peer Computing 2008 (P2P)*, pages 343–350, September 2008.
- [12] Sebastian Kaune, Jan Stolzenburg, Aleksandra Kovacevic, and Ralf Steinmetz. Understanding BitTorrent's Suitability in Various Applications and Environments. In *Proceedings of the 3rd International Multi-Conference on Computing in the Global Information Technology (ComP2P)*, pages 256–262, July 2008.
- [13] Kalman Graffi, Aleksandra Kovacevic, Nicolas Liebau, , and Ralf Steinmetz. From Cells to Organisms: Long-Term Guarantees on Service Provisioning in Peer-to-Peer Networks. In *Proceedings of the 8th ACM SIGAPP International Conference on New Technologies of Distributed Systems (NOTERE)*, pages 1–6, June 2008.
- [14] Konstantin Pussep, Matthias Weinert, Aleksandra Kovacevic, and Ralf Steinmetz. On NAT Traversal in Peer-to-Peer Applications. In *WETICE Collaborative Peer-to-Peer Systems Workshop (COPS'o8)*, June 2008.
- [15] Kalman Graffi, Sebastian Kaune, Konstantin Pussep, Aleksandra Kovacevic, and Ralf Steinmetz. Load Balancing for Multimedia Streaming in Heterogeneous Peer-to-Peer Systems. In *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, page 6, May 2008.

- [16] Patrick Mukherjee, Aleksandra Kovacevic, Michael Benz, and Andy Schürr. Towards a Peer-to-Peer Based Global Software Development Environment. In *Proceedings of the Software Engineering Conference SE2008*, pages 204–216, February 2008.
- [17] Patrick Mukherjee, Aleksandra Kovacevic, and Andy Schürr. Analysis of the Benefits the Peer-to-Peer Paradigm brings to Distributed Agile Software Development. In *Lecture Notes in Informatics (LNI) - Proceedings, Series of the Gesellschaft für Informatik (GI)*, volume P-122, pages 72–77, February 2008.
- [18] Kalman Graffi, Aleksandra Kovacevic, Kyra Wulffert, and Ralf Steinmetz. ECHoP2P: Emergency Call Handling over Peer-to-Peer Overlays. In *International Workshop on Peer-to-Peer Network Virtual Environments (P2P-NVE)*, pages 58–67, December 2007.
- [19] Kalman Graffi, Konstantin Pussep, Sebastian Kaune, Aleksandra Kovacevic, Nicolas Liebau, and Ralf Steinmetz. Overlay Bandwidth Management: Scheduling and Active Queue Management of Overlay Flows. In *Proceedings of IEEE Local Computer Networks (LCN)*, pages 334–342, October 2007.
- [20] Aleksandra Kovacevic, Nicolas Liebau, and Ralf Steinmetz. Globase.KOM - A P2P Overlay for Fully Retrievable Location-based Search. In *Proceedings of the 7th IEEE International Conference on Peer-to-Peer Computing (P2P)*, September 2007.
- [21] Nicolas Liebau, Oliver Heckmann, Aleksandra Kovacevic, Omid Tafreschi, Markus Fidler, Andreas Mauthe, and Ralf Steinmetz. A Secure Quorum Based Membership Mechanism for P2P Systems. In *Proceedings of the 12th Americas Conference on Information Systems (AMCIS)*, August 2006.
- [22] Oliver Heckmann, Marc Gomez Sanchis, Aleksandra Kovacevic, Nicolas Liebau, and Ralf Steinmetz. A Peer-to-Peer System for Location-based Services. In *Proceedings of the Peer-to-Peer Paradigm (PTPP) Track at AMCIS*, August 2006.
- [23] Nicolas Liebau, Oliver Heckmann, Aleksandra Kovacevic, Andreas Mauthe, and Ralf Steinmetz. Charging in Peer-to-Peer Systems based on a Token Accounting System. In *Proceedings of the 5th International Workshop on Advanced Internet Charging and QoS Technologies (ICQT)*, June 2006.
- [24] Sasa Rudan, Aleksandra Kovacevic, Charles Milligan, and Veljko Milutinovic. Data Assurance in Conventional File Systems. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, pages 317–325, January 2005.

TECHNICAL REPORTS

- [1] Kalman Graffi, Aleksandra Kovacevic, and Ralf Steinmetz. Towards an Information and Efficiency Management Architecture for Peer-to-Peer Systems based on Structured Overlays. Technical Report KOM-TR-2008-2, Multimedia Communications Lab KOM, Technische Universität Darmstadt, Merckstr. 25, 64283 Darmstadt, Germany, March 2008.

- [2] Konstantin Pussep, Sebastian Kaune, Christof Leng, Aleksandra Kovacevic, and Ralf Steinmetz. Impact of User Behavior Modeling on Evaluation of Peer-to-Peer Systems. Technical Report KOM-TR-2008-07, Multimedia Communications Lab KOM, Technische Universitaet Darmstadt, November 2008.

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe. Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt 2009